| | |
|---|---|
| **Titre:** Title: | Conception d'une architecture de gestion de réseaux à l'aide d'agents mobiles |
| **Auteur:** Author: | Jonathan Lefebvre |
| **Date:** | 2003 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Lefebvre, J. (2003). Conception d'une architecture de gestion de réseaux à l'aide d'agents mobiles [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6985/ |

UNIVERSITÉ DE MONTRÉAL

# CONCEPTION D'UNE ARCHITECTURE DE GESTION DE RÉSEAUX À L'AIDE D'AGENTS MOBILES

JONATHAN LEFEBVRE
DÉPARTEMENT DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
MAI 2003

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

CONCEPTION D'UNE ARCHITECTURE DE GESTION
DE RÉSEAUX À L'AIDE D'AGENTS MOBILES

présenté par : LEFEBVRE Jonathan
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen composé de :

Mme BOUCHENEB Hanifa, Doctorat, président
M. PIERRE Samuel, Ph.D., directeur et membre
M. CHAMBERLAND Steven, Ph.D., codirecteur et membre
M. QUINTERO Alejandro, Doctorat, membre

*À ma conjointe Céline, ton support et ta compréhension m'ont aidé à passer au travers mes études.*
*À ma fille Audrey-Marie, qui a été une inspiration tout au long de ma maîtrise.*

# REMERCIEMENTS

Je désire remercier toutes les personnes qui m'ont accordé leur soutien au cours de l'élaboration de ce mémoire. Particulièrement, je tiens à remercier mon directeur de recherche M. Samuel Pierre et mon co-directeur de recherche M. Steven Chamberland pour leurs conseils, leur écoute et leur contribution.

Je tiens à remercier tout le personnel du LARIM et de l'École qui m'a aidé dans la finalisation de mon projet de maîtrise.

Je désire aussi remercier ma conjointe ainsi que mes parents pour leur important appui lors de mes études. Leur contribution s'est avérée très importante pour passer au travers des divers obstacles pendant ma maîtrise.

# RÉSUMÉ

La gestion de réseaux est une discipline très importante. La croissance des réseaux et l'augmentation de leur complexité rendent cette gestion très difficile. Il est donc primordial d'adresser cette complexification avec des outils de gestion appropriés. De tels outils doivent être efficaces, automatiques et fiables. Ainsi, il devient possible de diminuer la charge de travail des opérateurs de ces réseaux. De plus, l'automatisation des tâches de gestion permet une réaction plus rapide aux changements dans le réseau, qu'ils soient volontaires ou non. De tels outils de gestion ne doivent pas nuire au fonctionnement normal du réseau. Ils sont donc contraints à des requis de performance.

Le présent mémoire apporte d'abord une architecture de gestion de réseaux dont les éléments centraux sont des agents mobiles. Avant l'élaboration de ces agents mobiles, plusieurs modules et outils doivent être disponibles. En effet, le but est de fournir aux opérateurs de réseaux des moyens simples pour automatiser la gestion du réseau. L'architecture présente donc des interfaces de gestion uniformisées pour permettre la création d'agents mobiles généraux. Elle demeure flexible en permettant l'accès aux outils de gestion déjà existants sans l'intermédiaire de ces interfaces. L'agent mobile général peut donc utiliser les moyens de gestion uniformes alors que l'agent mobile spécialisé utilisera des moyens de gestion plus spécifiques qui ne sont pas nécessairement disponibles via ces interfaces. Les interfaces de gestion sont, en réalité, un accès vers le code de gestion qui est inclu dans des agents stationnaires. Cette séparation du code permet la diminution de la taille des agents et leur simplification. Ces derniers accèdent à ce code à l'aide des interfaces de gestion par l'intermédiaire d'un proxy.

Pour continuer dans la voie de la flexibilité, l'architecture doit pouvoir gérer des composantes dans un réseau hétérogène réel. Cela implique que certaines composantes ne peuvent pas accueillir des agents mobiles. L'architecture prévoit donc des tables

d'associations entre composantes et stations de gestion. Les stations de gestion ont comme but précis de gérer une ou plusieurs composantes. L'affectation des composantes aux stations est effectuée en fonction de la proximité d'une station de gestion à une composante donnée.

La seconde partie de ce mémoire présente les agents mobiles qui ont été créés à l'aide des outils développés dans le cadre de l'architecture. Nous avons principalement créé deux agents mobiles de gestion chargés tous deux d'effectuer un diagnostic dans un réseau hétérogène. Le premier est effectivement chargé de visiter plusieurs composantes et de diagnostiquer des pannes, alors que le second aide l'agent mobile de diagnostic à trouver de nouvelles routes vers les composantes qui ne sont pas atteignables à cause d'une panne.

Ce duo d'agents mobiles a été comparé à l'agent mobile de diagnostic seul. Les résultats démontrent que le duo d'agents mobiles a réussi, dans 35% des tests, à trouver la cause précise d'une panne simple alors que l'agent mobile de diagnostic seul en était incapable. Les résultats demeurent les mêmes si l'on compare ce duo à l'agent stationnaire de diagnostic. Finalement, nous avons fait une comparaison entre l'agent de diagnostic stationnaire et mobile. Les résultats obtenus indiquent que l'approche de gestion à distance demeure encore plus performante que l'utilisation des agents mobiles. Cependant, si le trafic de gestion était plus élevé, les agents mobiles seraient probablement plus avantageux en terme de trafic généré.

L'architecture proposée a démontré son efficacité pour effectuer le diagnostic de panne. Il reste à vérifier ses capacités avec d'autres tâches de gestion et à améliorer ses performances.

# ABSTRACT

Network management in today's important networks is mandatory. This management can become difficult because of the constant increase in both size and complexity of networks. It is now a fact that network managers need better tools that are efficient, automated and reliable. With such tools, network managers use less time to search, repair and manage their network. This fact becomes important when time and human resources are limited. These tools must be fast and should not overload the network.

This work presents a network management framework based on mobile agents. Before the creation of these agents, some tools and modules must be available. The framework provides this with simple tools to quickly build mobile agent that automates network management tasks. The framework gives access to uniform management interfaces to help create simple and general mobile agents. It still allows flexibility by giving the choice to mobile agent builders to use these interfaces or not. In the later case, mobile agents use specialized tools and are called specialized agents. This ability is based on the fact that some network management functions may not be available through uniform interfaces, which give access to network management code that is not included in mobile agents for size optimization. The management code is instead placed in stationary agents that are accessed by using a proxy from the mobile agent to the stationary agent.

The framework is able to manage heterogeneous networks. Since some network devices cannot receive mobile agents, this implies that some devices are managed remotely. The framework provides association tables that link network devices to management stations. These stations may manage many network devices. The associations are based on the proximity of a management station to a given network device.

The second part of this work presents two mobile agents that were made to validate the framework. The first one task is to visit as many devices as possible to collect a series of facts about them. The facts collected are later used to find the source of a network failure. The second agent is used to help the first one to find an alternate route from one device to another. This second agent is called when the diagnostic mobile agent is unable to continue its investigation.

These two mobile agents have been compared to the diagnostic mobile agent alone. Results show that these two mobile agents found, in 35% of test cases, the exact cause of a single failure. The exact cause in these test cases was not found by the diagnostic mobile agent alone. This remained true when we compared the two agents with a remote management agent that could not move. We also did a performance review of the diagnostic mobile agent compared to its remote counterpart. Our results advantage the remote diagnostic agent in terms of traffic generated. But measurements give us some clues that this may not remain true when much more network management traffic is needed.

The framework demonstrates its efficiency to locate and find network failure causes. Its ability to handle other network management tasks still needs to be verified. Also, we could improve framework performance furthermore.

Les résultats démontrent que le duo d'agents mobiles a réussi, dans 50% des tests, à trouver la cause précise d'une panne simple alors que l'agent mobile de diagnostic seul en était incapable.

# TABLE DES MATIÈRES

# LISTE DES FIGURES

# LISTE DES TABLEAUX

# LISTE DES SIGLES ET ABRÉVIATIONS

ADVR – Agent-based Distance Vector Routing

AMG – Agent mobile de gestion

API – Application Programming Interface

ATM – Asynchronous Transfer Mode

CORBA – Common Object Request Broker Architecture

CSM – Customer-based IP Service Monitoring

DCOM – Distributed Component Object Model

DOS – Denial of Service

DVR – Distance Vector Routing

EQL – Event Query Language

FIPA – Foundation for Intelligent Physical Agents

GT-ITM – Georgia Tech Internetwork Topology Models

ICMP – Internet Control Message Protocol

IP – Internet Protocol

ISO – International Organization for Standardization

JAR – Java Archive

JVM – Java Virtual Machine

KQML – Knowledge Query and Manipulation Language

LAN – Local Area Network

LRU – Least Recently Used

MANET – Multi-hop Ad-hoc Wireless Network

MAP – Mobile Agent Platform

MCD – Mobile Code Daemon

MDM – Mobile Distributed Manager

MIB – Management Information Base

NS – Network Simulator

PDA – Personal Digital Assistant

PEnv – Phoenix Environment

PVC – Permanent Virtual Channel

RIP – Routing Information Protocol

RMI – Remote Method Invocation

RMON – Remote Monitoring

RPC – Remote Procedure Call

RSVP – Reservation Protocol

SANTA – Security Agents for Network Traffic Analysis

SNMP – Simple Network Management Protocol

SPARTA – Security Policy Adaptation Reinforced Through Agents

TCP – Transmission Control Protocol

TCP/IP – TCP over IP

UDP – User Datagram Protocol

VCI – Virtual Channel Identifier

VMC – Virtual Managed Component

VPI – Virtual Path Identifier

# CHAPITRE 1

# INTRODUCTION

Les premiers réseaux informatiques étaient simples, relativement petits et souvent construits avec des composantes homogènes. La gestion de ces réseaux demeurait relativement simple pour leurs utilisateurs. Avec la croissance et l'intégration rapide des réseaux informatiques à toutes sortes d'applications, il est devenu évident que des mécanismes de gestion plus efficaces devaient être élaborés. La compétition, le développement d'Internet, la multitude de services à offrir et l'arrivée de composantes plus performantes ont grandement compliqué la tâche des administrateurs de réseaux. De plus, la croissance rapide des réseaux pose plusieurs problèmes d'évolutivité. Les réseaux informatiques sont devenus vitaux. La performance et la fiabilité d'un réseau peuvent faire la différence entre une entreprise en plein essor et une autre en déclin. Malgré des années de développement, aucune solution miracle n'existe aujourd'hui pour gérer les réseaux efficacement et pour appréhender leur complexité. Il devient donc essentiel de simplifier les tâches de gestion pour permettre une gestion plus rapide, plus efficace et moins coûteuse, ce qui fait l'objet de ce mémoire. Dans ce chapitre d'introduction, nous présenterons les concepts de base qui serviront à élaborer les éléments de la problématique. Par la suite, nous préciserons les objectifs de recherche pour terminer avec une esquisse du plan du mémoire.

## 1.1 Définitions et concepts de base

La gestion de réseaux est une discipline englobant cinq domaines fonctionnels : la gestion de configuration, la gestion de fautes, la gestion de sécurité, la gestion de performance et la gestion de comptabilité. En d'autres mots, le gestionnaire de réseaux est responsable de vérifier l'état du réseau, de le contrôler en plus d'éviter et de corriger les problèmes matériels et logiciels. Il doit aussi éviter à tout prix les brèches de

sécurité, assurer la protection des données, mesurer et évaluer les performances du réseau, en planifier l'expansion et garder une trace de son utilisation pour des fins diverses telles que la facturation.

Il existe un certain nombre d'outils de gestion. Le principal est le protocole SNMP (Simple Network Management Protocol). SNMP est en fait bien plus qu'un protocole. Il constitue un modèle de gestion incluant un gestionnaire centralisé, des composantes matérielles ou logicielles à gérer exécutant un agent SNMP, une base de données de gestion nommée MIB (Management Information Base) et le protocole SNMP proprement dit. Ce dernier permet la saisie et la modification des paramètres des composantes du réseau. Ses principaux atouts sont sa simplicité et son omniprésence dans les composantes réseaux actuelles.

Le modèle RMON (Remote Monitoring) permet la distribution de l'intelligence de gestion en plusieurs sous-domaines de gestion. Il prévoit l'installation d'entités chargées de surveiller certains paramètres dans ces sous-domaines pour prendre des mesures plus précises. Ces mesures sont plus localisées et génèrent ainsi moins de trafic dans le réseau.

CORBA (Common Object Request Broker Architecture) et Java RMI (Remote Method Invocation) sont deux approches plus modernes qui permettent de distribuer la gestion sur plusieurs nœuds dans le réseau. Ils permettent tous deux d'exécuter des procédures de gestion à distance RPC (Remote Procedure Call). Contrairement à un protocole de gestion, ils permettent plus de flexibilité. En réalité, ces deux technologies ne sont pas exclusives à la gestion de réseaux, mais elles sont largement utilisées dans ce domaine. CORBA permet l'interopérabilité entre plusieurs langages de programmation, dont Java et C++, alors que Java RMI est limité à Java. Java est un langage de programmation qui est extrêmement portable d'une architecture à une autre et qui est devenu un incontournable dans les réseaux actuels.

L'intelligence de gestion qualifie la capacité d'une entité à effectuer une ou plusieurs tâches de gestion. Lorsqu'il est question de décentraliser l'intelligence de gestion, ces tâches de gestion peuvent être exécutées à plusieurs endroits dans le réseau.

En d'autres mots, la gestion du réseau se fait de manière répartie et parallèle sur diverses composantes.

Un réseau constitué de plusieurs composantes de différents manufacturiers ou utilisant différents protocoles est appelé *réseau hétérogène*. C'est ce type de réseau qui compose la majorité des installations actuelles. Un *réseau homogène* est, par opposition, construit avec des composantes provenant du même manufacturier et utilisant les mêmes protocoles. À cause des coûts, de la compétition, des différents services et besoins, de l'évolution et de la fusion de plusieurs réseaux, de tels réseaux sont très rares.

Dans ce mémoire, nous introduirons le concept d'agents mobiles pour gérer le réseau. Un agent mobile est un programme plus ou moins intelligent ayant la possibilité de se déplacer. Il utilise les ressources localisées à différents endroits pour mener à bien ses objectifs. Dans le cadre de ce mémoire, les objectifs de cet agent sont des tâches de gestion. L'agent mobile doit agir en grande partie sans l'intervention des utilisateurs du réseau. Lorsqu'il se déplace, il le fait avec son état d'exécution et sa mémoire. Un agent mobile est polymorphique dans le sens qu'il englobe plusieurs paradigmes de gestion énoncés jusqu'ici. Il peut agir à distance ou localement. Il peut se confondre avec une procédure à distance et peut communiquer avec le protocole SNMP. Il a l'avantage sur toutes ces méthodes de gestion de pouvoir se déplacer pour s'adapter ou pour répartir la charge sur plusieurs composantes dans le réseau. À première vue, il semble donc très bien adapté à la gestion de réseaux hétérogènes.

## 1.2 Éléments de la problématique

La gestion de réseaux est une discipline essentielle à tous réseaux informatiques. L'importance qu'on y accorde au sein d'un organisme devrait toujours être fonction de l'importance du réseau. La gestion a pour but de configurer, surveiller, analyser, modéliser, planifier, améliorer et réparer les réseaux informatiques. Plus le nombre de composantes et de services à gérer augmente, plus les probabilités sont élevées qu'une défaillance, une baisse de performance ou un problème quelconque

surviennent. Les tâches de gestion deviennent donc colossales à mesure que le réseau s'agrandit et se diversifie. Cette complexification est le résultat d'une perpétuelle évolution de la clientèle des réseaux et de leurs besoins. Malheureusement, les systèmes de gestion actuellement utilisés n'ont pas la capacité de suivre cette évolution avec élégance. Il devient donc primordial de considérer des solutions de gestion plus efficaces pour les réseaux hétérogènes de grande taille.

La quasi-totalité des réseaux actuels utilisent des systèmes de gestion centralisés. Ces systèmes de gestion, pour la majorité basés sur le modèle client-serveur de SNMP, posent des problèmes de taille pour les réseaux moindrement complexes. Ce modèle client-serveur fournit beaucoup trop d'informations et peu de moyens pour les gérer. Plusieurs travaux (Liotta et al., 2002; Rubinstein et al., 2000; Chang et Kao, 2001) dénoncent l'incapacité des systèmes de gestion centralisés à gérer efficacement un réseau hétérogène de grande taille. Ces systèmes centralisés étant très statiques, l'intelligence de gestion est centralisée ou faiblement répartie. Ce fait impose donc une charge de travail accrue sur les stations de gestion.

Le modèle centralisé utilise beaucoup de ressources du réseau, privant ainsi les utilisateurs d'une partie de ces ressources. Par exemple, une activité de surveillance à distance peut prendre une fraction de la capacité d'un lien. Plus le réseau est étendu, plus on doit multiplier les activités de surveillance. Avec cette multiplication, on génère plus de trafic, qui doit nécessairement traverser le réseau. Avec la somme de toutes les tâches de gestion, le trafic total peut réellement nuire à l'utilisation normale du réseau.

Les mesures de performance du réseau sont en quelque sorte erronées par l'approche client-serveur, car ces mesures génèrent du trafic non utile. Les stations de gestion deviennent aussi des goulots d'étranglement. L'ajout ou la désinstallation de certaines composantes peut devenir problématique à cause de la faible adaptabilité et la faible évolutivité des systèmes centralisés. L'approche client-serveur est donc très peu adaptée au concept de mobilité des composantes et des usagers. Elle requiert des liens fiables et de hautes capacités, alors qu'on voudrait pouvoir gérer des réseaux ayant des

composantes faiblement connectées. On voudrait pouvoir assurer la gestion même lors de partitionnement du réseau. Cela n'est pas évident avec le modèle client-serveur.

Plusieurs solutions déplaçant l'intelligence de gestion vers les nœuds à gérer ont été développées. On peut citer ici RMON, CORBA et Java RMI. Ces technologies adressent plutôt bien le problème posé par les approches centralisées, mais elles sont toujours très peu évolutives et consomment toutes beaucoup de ressources de réseau. Elles demeurent toujours largement dépendantes d'une entité centrale et du modèle client-serveur. Elles n'offrent pas la possibilité de s'adapter dynamiquement aux différentes modifications de charge ou de configuration du réseau et manquent donc de flexibilité.

Les gestionnaires du réseau font souvent face à des problèmes complexes. Peu de personnes ont véritablement l'expertise nécessaire pour gérer l'ensemble d'un réseau de grande taille. Aussi, peu de personnes ont une vision claire, précise et complète du réseau à gérer. Ce manque d'expertise exige un investissement énorme en temps. Le temps est malheureusement une ressource très limitée dans les réseaux. En effet, un réseau de téléphonie cellulaire ne peut se permettre d'avoir des défaillances ou des pertes de performance visibles à l'utilisateur. Aussi, un fournisseur de service Internet doit satisfaire ses clients, sans quoi ils iront vers la compétition. Plusieurs tâches de gestion sont simples et répétitives et exigent un déplacement d'un opérateur humain vers la composante à gérer. Ces tâches gagneraient à être automatisées et chaque déplacement évité est un gain de temps et d'argent.

Il devient donc de plus en plus évident que les tâches de gestion doivent être améliorées et automatisées. Pour pousser la gestion à un niveau plus élevé, les réseaux ont besoin d'un système capable d'appréhender leur complexité et leur évolution constante. Plusieurs travaux récents, que nous verrons dans le prochain chapitre, semblent indiquer que les agents mobiles pourraient être des outils inestimables pour mener à bien un certain nombre de tâches de gestion. Ils ont l'avantage d'être mobiles et peuvent donc facilement modifier leur comportement, leur nombre ou leur positionnement pour s'adapter aux conditions changeantes des réseaux. Ils ont le

potentiel d'équilibrer la charge sur le réseau par leur parallélisme d'exécution. Par leur mobilité, ils peuvent se positionner stratégiquement sur une composante pour recueillir de l'information plus précise, effectuer efficacement une tâche de gestion ou continuer une tâche de gestion pendant une déconnexion. Ils posent néanmoins un certain nombre de problèmes.

Il est donc de plus en plus évident qu'une solution de gestion à base d'agents mobiles devra s'intégrer aux autres infrastructures et développements existants. Cela implique des agents mobiles fonctionnant dans un environnement utilisant des technologies de gestion clients-serveurs. Cette intégration amène son lot de problèmes. Aussi, il existe beaucoup de modèles de gestion à base d'agents mobiles, mais peu d'applications réelles. Ces modèles rencontrent des problèmes d'implantation. Par exemple, les agents mobiles requièrent des environnements d'exécution impossibles à installer sur une multitude de composantes. Pour sa part, le modèle centralisé SNMP a l'avantage d'être simple à implanter. De plus, le transport de l'intelligence de gestion vers les nœuds à gérer cause des problèmes potentiels de sécurité. Ce transport peut aussi repousser certains manufacturiers qui construisent leurs composantes comme des boîtes noires. Finalement, les systèmes multi-agents mobiles engendrent plusieurs problèmes complexes tels que la coordination et la communication, tandis que les systèmes mono-agents, simples à implanter, offrent peu d'avantages en terme de performance (temps de réponse élevé).

## 1.3 Objectifs de recherche

L'objectif principal de ce mémoire est d'évaluer et de démontrer l'utilité des agents mobiles pour la gestion de réseaux en les intégrant à un contexte d'utilisation bien réel. L'ambition de ce projet n'est pas de concevoir un système de gestion exclusivement basé sur les agents mobiles. Le mémoire ne projette pas non plus d'implanter un système touchant à tous les domaines fonctionnels de la gestion de réseaux. Il vise plutôt leur utilisation pour effectuer efficacement un certain nombre de tâches de gestion. Il est donc question d'établir un système de gestion fonctionnel qui

utilise les agents mobiles pour des tâches de configuration, de sécurité, de gestion de fautes et d'analyse de performance. De manière plus spécifique, ce mémoire a pour objectifs de :

- concevoir et réaliser un système de gestion à base d'agents mobiles simplifiant, améliorant, accélérant et automatisant l'accomplissement de certaines tâches présentement effectuées à distance, inefficacement ou avec l'intervention d'un opérateur humain ;
- intégrer ce système de gestion à base d'agents mobiles à un réseau réel pour en évaluer les avantages et les inconvénients dans un véritable contexte d'utilisation ;
- évaluer et analyser la pertinence, l'efficacité et la performance du système proposé de gestion à base d'agents mobiles ;
- établir une architecture pouvant facilement être étendu pour l'ajout de tâches et de fonctionnalités de gestion dans des travaux futurs.

## 1.4 Plan du mémoire

Le mémoire comprend cinq chapitres. Le premier chapitre est l'introduction. Le chapitre suivant fait une revue de la gestion de réseaux, des agents mobiles et des recherches combinant ces deux concepts. Le troisième chapitre décrit en détail l'architecture de gestion à base d'agents mobiles proposée dans ce mémoire. Il fera état des choix, des particularités et des tâches de gestion qui seront implémentés dans le cadre de ce projet. Le quatrième chapitre porte sur l'évaluation et l'implémentation de cette architecture de gestion et fait la lumière sur la réelle utilité des agents mobiles. De plus, il analyse et évalue les performances de l'architecture en relation avec les approches de gestion classiques utilisant des gestionnaires à distance. Le chapitre cinq, en guise de conclusion, fait une synthèse des travaux qui ont été effectués, des résultats obtenus et donne un aperçu des limitations du mémoire et quelques pistes pour des travaux futurs.

# CHAPITRE 2

# GESTION DE RÉSEAUX À L'AIDE D'AGENTS MOBILES

La gestion de réseaux à l'aide d'agents mobiles est un domaine de recherche très actif. Beaucoup de recherches entrevoient ou montrent de l'intérêt pour les agents mobiles. Il est alors impossible d'envisager toutes les applications dans cet ouvrage. Néanmoins, nous avons pris soin de décrire les principaux travaux dans ce domaine ainsi que ceux qui pouvaient nous intéresser le plus dans nos propres recherches. Nous avons déjà introduit ce qu'était un agent mobile. Nous enchaînerons avec les avantages et les inconvénients des agents mobiles pour la gestion de réseaux. Nous donnerons des modèles généraux qui sont présentés dans la littérature. Cette introduction aux agents mobiles permettra de mieux comprendre la suite de ce chapitre. Il sera ensuite énuméré un certain nombre d'architectures académiques et commerciales. Finalement, nous donnerons des applications plus spécifiques aux cinq domaines fonctionnels de la gestion de réseaux définis par la classification ISO (International Organization for Standardization).

## 2.1 Synthèse des techniques de gestion actuelles

Dans cette section, nous explorerons les problématiques des différents systèmes et techniques de gestion. Nous verrons d'abord les systèmes de gestion centralisée suivis des solutions distribuées. Nous enchaînerons avec les méthodes utilisant les agents mobiles.

### 2.1.1 Problèmes des systèmes de gestion centralisés

Les technologies de gestion les plus employées dans les réseaux sont les approches client-serveur. La quasi-majorité de ces réseaux utilisent SNMP pour la gestion. Plusieurs recherches (Liotta et al., 1998; Rubinstein et al., 2000; Chang et

Kao, 2001) dénoncent l'incapacité des technologies de gestion SNMP de gérer efficacement les réseaux complexes, hétérogènes et à grande échelle. Le problème est que l'approche centralisée de SNMP, ou faiblement distribué de RMON, n'offre pas la possibilité de s'adapter aux différentes modifications de charge ou de configuration du réseau et manque donc de flexibilité. De plus, cette approche demande à l'administrateur un haut niveau de connaissance du réseau. Le modèle client-serveur fournit beaucoup trop d'informations et peu d'intelligence. Cela exige souvent aux opérateurs humains beaucoup de travail pour trouver la cause d'un problème (White et Pagurek, 1999).

La fusion des réseaux locaux et du réseau Internet ainsi que la disparition des réseaux ne comportant qu'un seul type de composante amène encore plus de complexité. Le gestionnaire central, ou les gestionnaires de domaine (approche distribuée statique), deviennent rapidement des goulots d'étranglement et sont peu tolérants aux fautes. De plus, le gestionnaire central contient toute l'« intelligence » et doit analyser seul toutes les données.

Le déploiement de nouvelles technologies ou de nouvelles procédures de gestion est très difficile étant donné la nature statique de ces solutions. Il faut aussi noter que toute la gestion se fait à distance en consommant des ressources du réseau et la bande passante. En effet, comme la majorité des tâches de gestion se fait par interrogation à distance (polling), la gestion ajoute constamment du trafic au réseau (Rubinstein et al., 2000). Cela est d'autant plus vrai quand un problème survient dans le réseau ou que ce dernier est congestionné (Puliafito et Tomarchio, 1999; Baldi et al., 1997). Dans ces situations, les opérateurs doivent sonder davantage le réseau, ce qui augmente encore plus le trafic et la congestion. La gestion du réseau avec SNMP et des gestionnaires distants contribue donc à sa dégradation. De toute évidence, toute activité de gestion aura un impact sur la performance du réseau, mais certaines approches sont moins invasives.

### 2.1.2 Problèmes des solutions distribuées

Liotta et al. (2002) ont indiqué que la distribution et la hiérarchisation sont des moyens de pallier certains problèmes énoncés précédemment. La distribution de gestionnaires de domaine est une approche utilisant ces moyens. C'est aussi la voie utilisée par RMON. D'autres formes de gestion décentralisée utilisent les technologies CORBA et Java RMI (Remote Method Invocation). Ces approches, quoique plus équilibrées en tant que répartition de la charge, ne fournissent pas la flexibilité recherchée. En effet, la décentralisation est statique et ne s'adapte pas aux changements sans interventions humaines. Ces approches ne permettent pas de s'adapter aux conditions changeantes dans un réseau.

### 2.1.3 Apport des agents mobiles à la gestion de réseaux

Beaucoup d'efforts et d'espoir ont été placés dans des solutions de gestion de réseaux à l'aide d'agents mobiles. La raison en est fort simple, les agents mobiles possèdent de nombreux avantages vis-à-vis des approches centralisées. De plus, selon Silva et al. (1999a), les agents mobiles ont le potentiel de régler tous les problèmes liés aux techniques de gestion conventionnelles.

Plusieurs des travaux portant sur l'intégration des agents mobiles à la gestion de réseaux manquent de résultats et certaines recherches démontrent que les agents mobiles ne sont pas toujours plus performants que certaines versions optimisées de système de gestion centralisée ou distribuée. Cependant, la gestion à l'aide d'agents mobiles présente certains avantages qu'aucune autre approche ne peut offrir.

#### *Avantages*

Sur les approches centralisées, les agents mobiles ont l'avantage de permettre d'équilibrer la charge de gestion en transportant l'« intelligence » près de l'information à traiter. Ainsi, un agent mobile peut se déplacer sur ou près d'un équipement du réseau pour obtenir et manipuler l'information nécessaire à la tâche de gestion. Si l'information doit être transportée, l'agent mobile peut réduire l'information sur place

en traitant l'information localement, diminuant ainsi la charge sur le réseau. Il peut aussi servir à surveiller ou interroger un équipement pour n'avertir le gestionnaire central qu'en cas de besoin. Il n'utilise donc le réseau qu'en cas de besoin, réduisant ainsi encore plus la charge sur le réseau. Toutes ces particularités permettent de réduire le trafic sur le réseau et la charge sur les gestionnaires centraux. Ces avantages sont d'ailleurs ceux qui sont les plus souvent énoncés dans les différents travaux.

La mobilité des agents mobiles permet aussi d'obtenir des informations de façon plus précise étant donné que le délai est minimal lorsque l'information est accédée localement (Kona et Xu, 2001; Bieszczad et al., 1998). Cette mobilité peut aussi permettre l'accès à certaines données difficiles d'accès à distance ou via des protocoles peu flexibles. La capacité de déployer des agents mobiles permet aussi une plus grande flexibilité quant à la mise à jour des tâches de gestion. En effet, changer une procédure de gestion devient fort simple avec les agents mobiles. Il suffit de rappeler les anciennes versions et de redistribuer les agents sur le réseau. Cette redistribution est moins évidente avec les agents stationnaires et pour les approches centralisées, car cela demande une réinstallation ou une redéfinition des protocoles et des composantes, ce qui est à toutes fins pratiques, soit difficile, impossible ou trop coûteux. Cette possibilité de redistribution et de mise à jour est parfois le seul argument qui justifie l'utilisation d'agents mobiles dans certaines applications de gestion. Les agents mobiles facilitent le déploiement ou le redéploiement de services et de logiciels dans le réseau. Ils peuvent aussi plus facilement choisir les emplacements pouvant bénéficier d'un logiciel ou d'un service et ceux qui n'en ont pas besoin, évitant ainsi une charge sur certains équipements du réseaux (La Corte et al., 2001).

Un autre avantage accordé aux agents mobiles est leur forte possibilité de s'adapter. En effet, certaines tâches du réseau peuvent être poursuivies en cas de problèmes, de partitions et de bris de connexions. Ils s'adaptent aussi aux modifications du réseau et à leur hétérogénéité. Les agents mobiles couplés à des agents stationnaires permettent de gérer avec élégance l'hétérogénéité caractéristique des réseaux modernes. De plus, ils peuvent se déplacer pour s'adapter à une charge et

peuvent réagir automatiquement à une modification de leur environnement (Liotta et al., 2002; Gavalas et al., 2002). Ils permettent le parallélisme d'exécution des tâches en se dispersant sur plusieurs nœuds sans consommer les ressources d'un gestionnaire central. C'est ce dynamisme qui avantage beaucoup les agents mobiles. La complexification des réseaux rend leur gestion compliquée. Les agents mobiles peuvent donc exécuter certaines tâches complexes et répétitives sans l'intervention d'un opérateur humain.

La technologie des agents mobiles s'adapte très bien aux réseaux ad hoc par la possibilité d'effectuer des tâches dans des conditions où il y a peu de connectivité. Qu'on soit connecté de manière intermittente ou par un lien de faible qualité ou capacité, l'agent mobile peut être lancé et retourné plus tard avec les informations recueillies.

Les agents mobiles ont aussi l'avantage de pouvoir s'intégrer avec les nombreuses autres technologies et architectures de gestion. Ainsi, un agent mobile peut utiliser CORBA ou Java RMI et peut même interagir avec les gestionnaires SNMP. Cela augmente davantage l'interopérabilité des agents mobiles et leur facilité d'adaptation aux environnements hétérogènes.

Avec un bon degré d'intelligence, les agents peuvent grandement contribuer à améliorer la rapidité avec laquelle les opérateurs humains peuvent mettre en place les réseaux et en solutionner divers problèmes.

***Inconvénients et limitations***

Un premier reproche aux systèmes à base d'agents mobiles, c'est qu'ils ne sont pas toujours plus performants que les systèmes de gestion à distance optimisés. Pour que la migration des agents mobiles n'apporte pas plus de trafic que les approches centralisées, il est nécessaire de garder le code de l'agent le plus petit possible. En effet, plus l'agent est intelligent, plus ce code est gros; et plus l'agent accumule de l'information en se déplaçant, plus ce code augmente. Aussi, l'accroissement des

ressources et des vitesses de transfert peut donner l'illusion que la gestion centralisée n'est pas si envahissante.

Les agents mobiles peuvent engendrer un temps de réponse plus long vis-à-vis les approches centralisées ou à agents stationnaires. Les recherches imputent ce fait à la quasi-obligation d'utiliser des langages interprétés qui sont lents (Onishi et al., 2001) ainsi qu'au temps de parcours du réseau des agents mobiles (Gavalas et al., 2002). Le langage interprété est important pour assurer une portabilité sur les réseaux hétérogènes (Susilo et al., 1998).

D'autres problèmes peuvent être matériels. Ce ne sont pas tous les équipements du réseau qui peuvent recevoir des agents mobiles. Ils ne sont pas tous prêts pour cette technologie. De plus, ce n'est pas tous les équipements du réseau qui ont avantage à exécuter des agents. Si on prend comme exemple les réseaux constitués d'un équipement léger tel que le PDA (Personal Digital Assistant), on peut se demander si l'exécution d'agents mobiles et de leur environnement d'exécution n'empêcherait pas le fonctionnement normal du PDA.

L'introduction d'agents mobiles amène de nouveaux problèmes de sécurité qui peuvent être inacceptables dans des réseaux où il y a trop de sources potentiellement hostiles. Divers hôtes pourraient facilement modifier le code d'un agent mobile ou encore altérer ses données. Ces mêmes hôtes pourraient lire de l'information qui ne leur ait pas destinée. D'un autre côté, les hôtes doivent aussi être protégées contre les agents mobiles hostiles. Ce problème est d'autant plus important si le réseau est connecté à Internet. De plus, plusieurs études démontrent que les attaques viennent souvent de l'interne. La sécurité est donc un motif majeur qui contribue à la faible acceptation des agents mobiles pour la gestion de réseaux.

La sécurité et divers autres problèmes empêchent encore d'utiliser les agents mobiles pour automatiser et améliorer les tâches de gestion d'un réseau. La complexité du paradigme de mobilité et d'intelligence mobile peuvent être difficile à appréhender et peuvent comporter des risques sur l'intégrité du réseau. Les agents mobiles sont décrits comme des entités pouvant agir sans l'intervention de l'utilisateur. Cette

automatisation peut comporter certains risques qui ajoutent à la complexité, donc à la difficulté de les intégrer aux réseaux courants.

Les communications entre agents mobiles demeurent complexes. Les systèmes actuels à base d'agents mobiles se limitent souvent à un seul agent ou à plusieurs agents ayant peu, sinon aucune interaction entre eux.

***Atténuation des inconvénients***

Il est intéressant de noter qu'avec les nombreuses recherches sur la gestion de réseaux à l'aide d'agents mobiles, les inconvénients en viennent à s'estomper. Le temps de réponse peut devenir meilleur que les systèmes de gestion SNMP centralisés avec des modifications intelligentes. Les problèmes de sécurité font l'objet d'une attention importante de la part des développeurs de plates-formes d'agents mobiles. Pour gérer la complexité, plusieurs outils et mécanismes de gestion des plates-formes ont été étudiés. Pour ce qui est des équipements non aptes à exécuter des agents mobiles, on a introduit la notion de proximité (Pinheiro et al., 1999). De plus, selon White et al. (1999), les manufacturiers d'équipements vont éventuellement offrir des plates-formes d'exécution Java dans une majorité d'équipements réseaux.

Comme les agents mobiles peuvent prendre diverses formes, il est simple d'améliorer les modèles en limitant ou augmentant la mobilité de certains agents, en augmentant le nombre d'agents ou en les clonant. Il est alors possible d'implémenter des systèmes utilisant les ressources du réseau intelligemment et plus efficacement que les approches clients-serveurs optimisées (Liotta et al., 2002).

## 2.2 Modèles de gestion de réseaux à base d'agents mobiles

Avant de présenter les diverses architectures disponibles sur le marché ou dans les domaines de recherches, il convient de synthétiser les principaux modèles existants suggérés dans la littérature. Dans cette section, nous donnons un bref aperçu des diverses utilisations des agents mobiles qui reviennent le plus souvent.

### 2.2.1 Mono-agent

Le modèle avec un seul agent demeure le modèle le plus simple, car il n'exige aucune communication et l'intelligence requise est incluse dans un seul agent. Cependant, ce modèle ne donne pas de bons résultats dans les réseaux avec un très grand nombre d'éléments à gérer. L'équipe de Rubinstein et al. (2002) a effectué une étude qui démontre qu'un seul agent visitant chaque nœud à gérer donne un temps de réponse plus élevé qu'une approche client-serveur. De plus, selon Gavalas et al. (2002) et Liotta et al. (2002), une approche multi-agents où plusieurs agents ou gestionnaires se séparent le réseau donne des résultats beaucoup plus intéressants. Le modèle mono-agent demeure néanmoins une solution intéressante dans certains cas où le temps de réponse n'est pas un problème. Il peut diminuer la complexité des communications.

### 2.2.2 Multi-agents

Les modèles multi-agents sont majoritaires dans la littérature et prennent plusieurs visages. Certains modèles présentent plusieurs agents ayant chacun une spécialisation propre. Dans un tel modèle, les agents communiquent entre eux et ne font appel à certaines spécialités qu'en cas de besoin. D'autres entendent par multi-agents un modèle où plusieurs agents mobiles identiques se séparent les éléments à gérer en différents domaines.

Dans les systèmes multi-agents, la communication inter-agents est soit directe (communication avec un agent spécifique) ou indirecte (traces, messages laissés sur un nœud sans destinataire propre). Les agents doivent coopérer avec les autres agents pour atteindre leurs buts et échangent des messages en respectant et spécifiant certaines ontologies. La coopération implique un certain synchronisme. L'ontologie, dans ce contexte, est une collection de termes et de règles définissant et gouvernant un certain domaine (Bieszczad et al., 1999). Évidemment, tout ceci introduit une certaine complexité aux systèmes multi-agents.

### 2.2.3 Distribution des agents mobiles et des gestionnaires

Pour améliorer les performances des systèmes de gestion d'agents mobiles et pour les adapter à toutes sortes de réseaux, plusieurs approches dans leur distribution ont été élaborées. Ainsi, on a les systèmes où plusieurs agents sont plutôt libres de migrer n'importe où selon des fonctions déterministes simples ou des fonctions purement aléatoires. D'autres systèmes introduisent l'idée de séparer les éléments à gérer en plusieurs domaines fixes. Chaque domaine contient un gestionnaire statique contrôlé par un gestionnaire central qui peut lancer des agents mobiles dans son domaine. Pour pousser plus loin encore cette approche, certains ont eu l'idée d'implémenter ces gestionnaires de domaines comme étant des agents mobiles qui, au besoin, se déplacent dans leurs domaines pour s'adapter à la charge des équipements (Gavalas et al., 2002). Ces agents mobiles de gestion peuvent lancer des agents mobiles spécialisés pour la gestion de leur nouveau domaine. Dans certains cas, un domaine n'est pas nécessairement borné par un routeur ou un commutateur. Il peut être déterminé de toutes sortes de façons avec des positionnements et des partitionnements optimaux (Liotta et al., 2002) ou des fonctions de proximité (Sugar et Imre, 2001).

### 2.2.4 Types de mobilité des agents

Les agents mobiles peuvent être déployés de plusieurs façons. Pour les systèmes où l'on s'intéresse particulièrement à surveiller des éléments de réseaux et où l'on fait des mises à jour fréquemment, on peut n'être intéressé que par leur possibilité d'aller vers les éléments à gérer (Get n' Stay de Gavalas et al. (2000)). Ainsi, l'agent ne migre qu'une seule fois puis s'éteint lors d'une mise à jour. D'autres approches visent à envoyer un agent par nœud (diffusion ou « broadcast » en anglais) pour gérer et recueillir les résultats à leur retour au gestionnaire central (Kona et Xu, 2001). C'est en quelque sorte un allez retour. Finalement, il y a l'approche où chaque agent visite plusieurs nœuds avant de revenir avec l'information recueillie au gestionnaire central ou de domaine (Get n' Go de Gavalas et al. (2000)). On parle alors d'agents

comportant un itinéraire et un plan de tâches à effectuer. Certaines architectures combinent plusieurs de ces types de mobilité.

### 2.2.5 Interopérabilité avec les systèmes de gestion existants

L'interopérabilité devient un facteur déterminant dans les recherches récentes. Les chercheurs ont en effet réalisé qu'il était impossible, pour l'instant, de développer un système de gestion à l'aide d'agents mobiles qui ne tient pas compte des protocoles et systèmes de gestion existants (Silva et al., 1999b). C'est pourquoi les architectures tendent à intégrer le standard MASIF, qui entre autres définit une interface permettant l'échange des agents mobiles entre plates-formes d'agents mobiles et l'intégration de CORBA. De plus, la majorité des recherches soutiennent l'importance d'intégrer, d'une manière ou d'une autre, SNMP aux systèmes de gestion à l'aide d'agents mobiles. C'est en fait le seul moyen de mettre en œuvre aujourd'hui un système de gestion à base d'agents mobiles. De plus, cette approche permet l'intégration progressive des agents mobiles sans devoir complètement laisser tomber les approches clients-serveurs.

## 2.3 Architectures et plates-formes d'agents mobiles

Dans cette section, nous présentons les diverses architectures d'agents mobiles. En premier lieu, nous présentons des architectures dont l'objectif principal est la gestion de réseaux. Ensuite, il sera question d'un survol sommaire de quelques plates-formes commerciales qui ne sont pas destinées à une application précise.

### 2.3.1 Architectures dédiées à la gestion de réseaux

Plusieurs architectures de gestion de réseaux à base d'agents mobiles ont été élaborées dans des contextes académiques ou privés. Il est impossible de lister chacune d'entre elles, mais cette section présentera quelques-unes de ces architectures.

### *PERPETUUM*

Plusieurs chercheurs du « Network Management and Artificial Intelligence Laboratory » de l'université de Carleton ont développé un système de gestion de réseaux à base d'agents mobiles (Boyer et al., 1999; Susilo et al., 1998). Leur système envisage un avenir où une majorité de composantes réseaux auront la possibilité d'exécuter une machine virtuelle Java nommée JVM (Java Virtual Machine). L'infrastructure mise en place prévoit l'installation d'une JVM sur chaque équipement devant être géré. Cette machine virtuelle doit exécuter un environnement appelé MCD (Mobile Code Daemon) permettant l'exécution des agents mobiles. Le MCD alloue un fil (thread) à chaque agent mobile dans la machine virtuelle et donne accès à divers services tels qu'un service de migration et de communication. Pour assurer l'interopérabilité et éviter aux agents mobiles de devoir maintenir des procédures d'accès différentes pour tous les équipements, on dote chaque élément à gérer d'une entité nommée VMC (Virtual Managed Component). Le VMC permet de créer une interface commune à tous les éléments pour simplifier le programme des agents mobiles. Par exemple, c'est par le VMC qu'un agent peut avoir accès à des informations contenues dans une MIB de SNMP. Chaque VMC peut donc utiliser des stratégies différentes pour interfacer l'agent mobile et l'équipement.

Pour des raisons de sécurité, les agents n'ont jamais la permission d'accéder aux ressources des équipements directement. Ils doivent toujours passer par le VMC et être authentifiés par la JVM avant leur exécution. Pour la communication entre agents, le système prévoit un DAS (Directory Assistant Server) chargé de prendre note des déplacements des agents mobiles.

### *Gestion de réseaux avec MAP*

MAP (Mobile Agent Platform) est une plate-forme permettant le développement et la gestion d'agents mobiles. Elle a été construite à l'Université de Catania en Italie. Puliafito et Tomarchio (2000) ont décri un système de gestion à base d'agents mobiles utilisant MAP. L'idée est de placer un agent mobile sur chaque élément à gérer.

L'intelligence nécessaire est transférée pour mener à bien une tâche précise et pour prendre des décisions localement. Par exemple, une tâche de surveillance de certaines valeurs pourrait être assignée à un agent. Si un agent requiert plus d'intelligence, il en fait la demande au serveur et un agent spécialisé est envoyé. Puliafito et Tomarchio (2000) veulent ainsi limiter la quantité d'informations inutiles qui est transférée sur le réseau. Le système de gestion proposé fait une distinction entre l'auteur d'un agent mobile et son utilisateur. Cette distinction est importante pour connaître les responsables de certaines actions. Pour interagir avec les composantes utilisant SNMP, Puliafito et Tomarchio (2000) ont utilisé les classes Java de gestion de ADVENT.

***Phoenix***

Phoenix est une architecture pour la gestion de réseaux programmables utilisant les agents mobiles (Putzolu et al., 2000). Elle dispose de mécanismes de sécurité. Un agent mobile est composé de trois éléments : un itinéraire, une feuille de travail décrivant ce qu'il doit faire sur chaque nœud et une politique qui décrit ce qu'il doit faire en cas d'erreur ou d'exception. L'espace d'exécution des agents mobiles est appelé PEnv (Phoenix Environment). Lors de l'arrivée d'un agent dans un PEnv, il passe une série de vérifications de sécurité et demande s'il est possible de s'exécuter sur ce nœud. Si c'est le cas, il acquiert les ressources nécessaires pour sa tâche. Sinon, il est libre de se déplacer vers un autre nœud et réagit en fonction de ses politiques. Putzolu et al. (2000) indiquent que Phoenix a été utilisé pour faire de l'analyse de congestion et de la détection d'intrusion.

***JAMES***

JAMES est une autre plate-forme permettant la gestion de réseaux à l'aide d'agents mobiles (Silva et al., 1999b). Les motivations des auteurs de JAMES sont d'en faire une plate-forme plus efficace que les autres solutions disponibles. Elle est vraiment dédiée aux réseaux de télécommunications et à la gestion de réseaux. Silva et al. (1999b) veulent obtenir une haute performance, un système tolérant aux fautes, du

support pour la gestion de réseaux, une facilité de mise à jour, des mécanismes de contrôle de ressources, une interface de programmation simple, du support pour CORBA et une implémentation 100% Java. L'architecture possède une agence JAMES et un gestionnaire JAMES. L'agence est le lieu d'exécution des agents mobiles. Il dispose de plusieurs services permettant à l'agent d'accomplir sa tâche. Un agent mobile est doté d'un itinéraire et d'une mission. Le gestionnaire est le point central du système d'où les agents JAMES sont lancés. Chaque élément du réseau possède une machine virtuelle pour exécuter les agents mobiles.

JAMES innove dans sa façon de répartir et de conserver le code utile aux agents mobiles. Il dispose d'un serveur de code comportant un fichier JAR (Java Archive) par agents. Chaque agence possède son propre répertoire de code, ainsi qu'un cache LRU (Least Recently Used) permettant de conserver le code des agents les plus récemment exécutés. Ainsi, le système peut limiter les transferts si certains nœuds possèdent déjà l'agent dans leur cache ou leur répertoire. Pour être tolérantes aux fautes, les agences sauvegardent périodiquement leur état et les agents mobiles sont sauvegardés avec leur état avant chaque migration. Aussi, le gestionnaire peut informer toutes les agences sur l'itinéraire d'un agent pour qu'elles se préparent à son exécution en préchargeant le code nécessaire à cet agent.

JAMES permet l'ajout de SNMP à son système d'agents mobiles. Tout ajout peut se faire dynamiquement et peut facilement être enlevé. Les auteurs de JAMES affirment qu'il est impossible d'éliminer l'utilisation des protocoles. C'est pourquoi JAMES a été construit avec la possibilité d'ajouter des modules permettant l'interopérabilité avec les standards. JAMES peut donc interagir avec les nœuds utilisant SNMP, et peut aussi être géré par les gestionnaires SNMP.

### *Autres architectures dédiées*

D'autres plates-formes existent. On peut nommer ici INCA, Discovery, Magenta et LucINA. Elles offrent toutes des particularités différentes selon leurs objectifs. Cependant, il semble de moins en moins intéressant de se tourner vers des

architectures développées dans un cadre académique ou privé. Dans les travaux les plus récents, les architectures commerciales sont de plus en plus utilisées car elles sont devenues plus stables et contiennent tous les outils nécessaires à la construction des agents. Ces architectures commerciales permettent aux chercheurs de développer des applications sans avoir à développer une infrastructure, ce qui accélère grandement la production de systèmes à base d'agents mobiles.

### 2.3.2 Architectures commerciales

Dans cette section, nous faisons une brève description des architectures commerciales. Ce domaine est en perpétuelle expansion et les diverses architectures tendent à offrir les mêmes fonctionnalités. Les informations sur ces architectures nous proviennent du site Internet de AgentBuilder.

#### *Aglets*

Aglets a été développé par une division japonaise de IBM. Un aglet est un objet Java qui peut se déplacer d'hôte en hôte en conservant son état d'exécution et sa mémoire. Il est probablement l'outil commercial le plus populaire et la plate-forme offerte est très robuste.

#### *Voyager*

Voyager est développé par la compagnie « Object Space ». Il combine les agents mobiles avec la possibilité d'utiliser les invocations à distance de CORBA. Cela permet donc de développer facilement des applications utilisant les agents mobiles sans modifier complètement les infrastructures. Voyager offre des agents mobiles qui conservent leur état d'exécution et leur mémoire.

#### *Grasshopper*

Grasshopper a été conçu par la compagnie allemande IKV++. Cette plate-forme est compatible avec le standard MASIF et une partie de FIPA (Foundation for

Intelligent Physical Agents). Il peut intégrer CORBA et est implémenté complètement en Java. Grasshopper offre aussi des agents mobiles qui conservent leur état d'exécution et leur mémoire. La plate-forme est simple à utiliser.

## 2.4 Gestion de configurations

La gestion de configuration est un élément important de la gestion de réseaux. La gestion de réseaux, autant pour un opérateur humain qu'un programme intelligent, ne peut être faite sans avoir une vue du réseau à gérer. La gestion de configuration permet de découvrir le réseau en observant son état et en dénichant ses composantes. Elle est donc une première étape essentielle pour la gestion à l'aide d'agents mobiles, car elle permet aux agents comme aux opérateurs de modéliser le réseau (Lazar et al., 1999).

Nous verrons donc dans un premier temps les recherches effectuées sur la modélisation du réseau à l'aide d'agents mobiles. Ensuite, nous verrons des applications pour les agents mobiles pour d'autres fonctions de la gestion de configuration.

### 2.4.1 Modélisation de réseaux et inventaire

Les motivations pour utiliser les agents mobiles pour la modélisation d'un réseau sont nombreuses. Ils permettent au gestionnaire de déléguer la tâche complexe de découverte aux éléments du réseau et de réduire l'utilisation des liens de communications. De plus, un modèle utilisant les agents mobiles sera vraisemblablement plus flexible et plus facilement extensible en incorporant les nouveaux éléments et services sans aucune intervention ou configuration. Le mécanisme de découverte peut être modifié à tout moment en modifiant simplement les agents de découvertes.

Selon White et al. (1999), les outils de modélisation ne sont généralement destinés qu'à un certain nombre d'éléments du réseau. L'opérateur doit utiliser plusieurs de ces outils pour avoir une modélisation juste du réseau. White et al. (1999)

utilisent le système élaboré à l'Université de Carleton à Ottawa, soit un agent mobile qu'ils injectent dans le réseau pour découvrir les stations présentes ainsi que les services et propriétés intéressantes. Les agents mobiles peuvent interagir avec tous les équipements grâce aux VMC devant être installés sur toutes les composantes. L'agent peut être équipé de filtres permettant à la recherche d'inclure ou d'exclure certains éléments. White et al. (1999) vont même jusqu'à indiquer que les agents de découverte pourraient lancer un autre agent à partir de l'équipement nouvellement découvert pour aller chercher sur le site du manufacturier des informations sur l'équipement en question. Ces travaux ne sont pas appuyés par des résultats concrets et demeurent donc en grande partie des modèles.

Une autre étude moins ambitieuse, mais plus réaliste, permet la découverte de liens et de nœuds dans un réseau ATM (Asynchronous Transfer Mode) (Lazar et al., 1999). Elle propose un algorithme simple qui consiste à envoyer un agent sur chaque lien partant de la station de départ. Ensuite, si l'agent se retrouve sur un commutateur, il envoie une copie de lui-même sur chaque interface en se clonant puis retourne après son voyage avec l'information recueillie. Lorsqu'un clone se retrouve sur un élément autre qu'un commutateur, il s'éteint. La découverte, une fois initiée, est relancée dynamiquement pour découvrir les nouveaux éléments. Les agents peuvent poursuivre leurs recherches même en présence de partitions temporaires et continuer à offrir une vue maximale du réseau.

La modélisation de réseau à l'aide d'agents mobiles peut donc être utilisée à la fois pour donner une représentation logique et physique aux opérateurs humains, mais aussi pour concevoir et planifier automatiquement des itinéraires pour d'autres types d'agents mobiles.

### 2.4.2 Découvertes des ressources dans un réseau ad hoc

Dans un réseau ad hoc, des composantes peuvent être ajoutées, modifiées ou enlevées à tout moment. Pour utiliser de façon maximale les ressources d'un tel réseau, il est important de disposer d'un moyen de découverte qui s'accommode bien d'un

environnement changeant, sans fil et peu fiable, car on ne peut jamais garantir la disponibilité d'une ressource. L'idée est d'utiliser un agent mobile pour se déplacer de nœuds en nœuds pour obtenir de l'information sur les ressources locales. Pour accélérer les échanges d'informations, plusieurs recherches (Chpudhury et al., 2000; Storey et Blair, 2000) permettent aux agents mobiles d'utiliser l'information déjà découverte par les autres nœuds. Chpudhury et al. (2000) ont introduit le concept de vieillissement de l'information à l'aide de compteurs pour éviter que les agents obtiennent de l'information trop âgée. Storey et Blair (2000) considèrent quant à eux que les technologies telles que CORBA et DCOM (Distributed Component Object Model) ne rencontrent pas les requis d'un réseau sans fil. Ils proposent l'utilisation de « tuples spaces » et d'agents mobiles pour découvrir, emmagasiner et gérer l'information sur les ressources du réseau ad hoc. Ces « tuples spaces » permettent donc d'optimiser le processus de mise à jour.

### 2.4.3 Gestion de connexions

La gestion de connexions est une des tâches majeures des réseaux ATM. Comme chaque manufacturier offre des méthodes différentes pour configurer et contrôler des connexions permanentes virtuelles (PVC, Permanent Virtual Connection), il devient très compliqué d'automatiser le processus (Boyer et al., 1999; Pagurek et al., 1998). Selon Cheikhrouhou et al. (2000), la configuration est compliquée étant donné un certain nombre de contraintes pour les PVC. Ces contraintes peuvent être de type architectural tel que l'importance d'avoir le même VPI (Virtual Path Identifier) et VCI (Virtual Channel Identifier) de la sortie d'un commutateur à l'entrée d'un autre. De plus, même lorsqu'un manufacturier supporte SNMP, les MIB peuvent tout de même être différentes. La configuration manuelle peut mener à des erreurs difficiles à détecter.

Boyer et al. (1999) ont proposé de simplifier la gestion en utilisant l'architecture et les concepts développés dans le cadre du projet PERPETUUM. Avec l'utilisation d'une interface VMC, ils peuvent uniformiser l'accès aux différentes informations

nécessaires à la configuration de connexions PVC et en automatiser la gestion. Les agents mobiles sont dotés d'une certaine autonomie pour leur permettre de mener à bien leurs tâches de configuration et de gérer les nombreuses contraintes. Ils envisagent deux stratégies pour permettre la configuration. La première est de lancer un agent qui établit les connexions PVC d'un nœud à un autre et qui retourne sur ses pas lors d'une erreur. Cet agent transporte toute l'intelligence ainsi que les requis nécessaires à l'établissement d'une liaison de bout en bout (Boyer et al., 1999; Pagurek et al., 1998). La seconde approche implique plusieurs agents mobiles envoyés à chacun des nœuds qui négocient entre eux pour établir une connexion. Ces agents peuvent avoir différentes tâches à chaque nœud. L'expérimentation montre que l'approche à un seul agent est environ cinq à six fois plus rapide autant pour la création de la connexion et son entretien que pour son relâchement. Les auteurs attribuent cet écart au besoin de localiser l'agent avant d'établir une communication et aux délais de communications ordinaires.

### 2.4.4 Gestion de la qualité de service

Les besoins pour la qualité de service se font de plus en plus ressentir. Avec l'émergence des applications multimédias, la vidéo conférence et la téléphonie Internet, il devient important d'assurer un certain niveau de qualité.

La qualité de service relève en partie de la gestion de configuration. Cependant, nous aborderons cet aspect dans la gestion de performance étant donné les liens étroits qui existent entre la qualité de service et l'amélioration de la performance.

### 2.4.5 Routage

Dans cette section, nous présentons deux types de techniques de routage à base d'agents mobiles : une basée sur les colonies de fourmis et une amélioration du DVR (Distance Vector Routing).

### *Colonie de fourmis*

L'aptitude des fourmis à trouver le meilleur chemin a inspiré un certain nombre de recherches. Bien que ces recherches partent d'une prémisse simple, les fourmis et leur intelligence limitée, il n'en demeure pas moins que leur application dans le routage et leur effet global demeure complexe.

Schoonderwoerd et al. (1997) ont introduit l'idée d'utiliser des agents mobiles pour balancer la répartition des appels dans un réseau de télécommunications soit en plaçant mieux les appels ou soit en les redirigeant. Leur motivation pour utiliser l'idée des fourmis se base sur les observations. En effet, une colonie de fourmis peut fort bien continuer ses opérations en cas de perte d'une partie de ses membres. Cette propriété amène de la robustesse au système. Cela introduit un autre avantage. Les agents mobiles copiant les fourmis demeurent simples et légers et sont ainsi moins complexes et causent moins d'impact sur le réseau lors de leur migration.

Les agents mobiles calquant les fourmis se promènent dans le réseau en réagissant aux traces laissées par les autres agents et laissent leurs propres traces. Ils utilisent et modifient les tables de routage pour arriver à leurs fins et pour laisser des traces de leur passage. Les communications peuvent donc utiliser ces tables de routage dynamiques pour choisir une meilleure route. Les traces sont appelées phéromones, du même nom que l'hormone des vraies fourmis. C'est une forme de communication indirecte et coopérative, où chaque agent augmente l'intensité d'un message laissé aux autres. Un agent empruntera une route où il y a plus de phéromones. Pour avoir plus de phéromones, il faut qu'un maximum de fourmis emprunte le chemin. On peut donc s'attendre à ce que le chemin le plus avantageux (et non pas nécessairement le plus court) augmente rapidement sa concentration de phéromones. S'il y a congestion, les phéromones sont mises à jour moins souvent, permettant aux agents de choisir de nouvelles routes pour s'adapter.

Schoonderwoerd et al. (1997) ont intégré plusieurs retouches au fonctionnement de ce mécanisme de mise à jour du routage pour éviter des fonctionnements pathologiques tels que des problèmes qui forceraient les agents à toujours choisir le

même chemin. Ces retouches sont l'introduction de mécanismes d'évaporation et la possibilité qu'une fourmi ne choisisse pas toujours le meilleur chemin. Ils veulent ainsi s'assurer de toujours obtenir la meilleure route. Cet algorithme a montré qu'il donnait de meilleurs résultats que l'utilisation d'un routage fixe utilisant le plus court chemin. Il fonctionne aussi mieux qu'un système proposé par Appleby et Steward (1994) utilisant deux types d'agents mobiles.

Di Caro et Dorigo (1998) et Barán et Sosa (2000) présentent des approches similaires nommées respectivement AntNet 1.0 et AntNet 1.1.

***Introduction d'agents mobiles dans le « Distance Vector Routing »***

Amin et al. (2001) ont tenté d'améliorer un algorithme de routage existant en utilisant les agents mobiles. Selon Amin et al. (2001), la version actuelle du « Distance Vector Routing », nommé DVR, crée un grand nombre de messages de mise à jour. DVR utilise la distance comme métrique et utilise une fonction simple pour mettre à jour les tables de routage des différents nœuds. La version modifiée, nommée ADVR pour « Agent-Based Distance Vector Routing », solutionne ce problème en limitant le nombre de messages échangés. Ce nombre est borné par le nombre d'agents dans le réseau. Dans ADVR, les agents visitent les nœuds adjacents et mettent à jour la table de routage selon une équation légèrement différente que pour le DVR. La mise à jour se fait moins fréquemment. La stratégie de migration est choisie minutieusement, car une mauvaise stratégie peut facilement dégrader la performance de ADVR. Deux stratégies sont énoncées : la marche aléatoire et la marche structurée. Dans la marche aléatoire, l'agent choisit un nœud adjacent au hasard. Pour la marche structurée, l'agent se déplace selon un modèle déterministe. Cette stratégie peut utiliser différents paramètres pour justifier le choix d'un nœud. Dans leurs travaux, Amin et al. (2001) suggèrent d'utiliser le nombre de visites. Ce choix est fait sur l'hypothèse que les nœuds les moins visités ont les tables de routage les moins à jour. Le décompte de visites peut se faire sur les nœuds, sur les liens, ou sur les deux à la fois.

La convergence pour trouver les meilleures routes avec ADVR est d'autant plus rapide que le nombre d'agents est grand. Cependant, si ce nombre est trop grand, les agents se mettent à causer un trop gros impact sur les ressources du réseau. L'avantage principal de ADVR, c'est qu'il échange beaucoup moins de messages et qu'il devient ainsi plus intéressant pour un réseau sans fil.

### 2.4.6 Routage dans un réseau ad hoc

Selon Onishi et al. (2001), le routage est nécessaire dans un réseau ad hoc pour étendre le champ d'action du réseau pour des équipements avec des ondes radios de courtes portées. Ils donnent comme exemple l'établissement d'un réseau sans fil dans une maison. Pour élaborer un réseau d'un bout à l'autre, il se peut fort bien qu'un message doive sauter d'une machine à une autre pour arriver à destination. Ils proposent l'utilisation des agents mobiles pour le routage dynamique. La solution proposée s'inscrit dans le contexte de MANET (Multi-hop Ad-hoc Wireless Network) et utilise un mécanisme de routage proactif, où les tables de routage sont mises à jour périodiquement. Ils utilisent des agents mobiles de routage nommés explorateurs qui sont envoyés pour découvrir les routes. Dans ce modèle, les tables de routage contiennent plusieurs entrées correspondant aux N derniers agents explorateurs arrivés. Chaque entrée de cette table agrandie inscrit le temps d'arrivée pour évaluer la « fraîcheur » de l'information ainsi que le nombre de sauts qui ont été nécessaires pour arriver à la destination. Les agents messagers, qui transportent de l'information utile, empruntent un chemin basé sur une fonction d'évaluation du meilleur chemin. Généralement, ce chemin correspond à un compromis entre l'âge de l'information et un nombre minimal de sauts.

Sugar et Imre (2001) ont indiqué qu'un routage dynamique est nécessaire dans les réseaux ad hoc car les liens ne sont pas fiables et une station servant au routage peut disparaître à tout moment. Ils proposent une solution à la fois proactive et réactive, étant donné qu'ils sont d'avis qu'une solution complètement proactive pollue le réseau. Ils introduisent le concept de grappes, où n'importe quel ordinateur d'une grappe n'est

jamais à plus de deux sauts d'un autre. Chaque grappe a un chef qui se met à jour périodiquement avec les autres chefs de grappes adjacentes. Pour trouver un chef de grappe, un agent mobile peut être déployé par la machine nouvellement entrée dans le réseau pour découvrir un chef de grappe dans son voisinage (de 2 sauts). Les agents mobiles permettent aussi plusieurs tâches de découvertes pour pouvoir recomposer dynamiquement des grappes pour s'assurer d'une connectivité maximale. Chaque paquet sortant et entrant dans une grappe passe par ce chef de grappe. Bien que cela introduise un certain potentiel pour un goulot d'étranglement, on constate une réduction des paquets de contrôle. Selon Sugar et Imre (2001), cette réduction est plus souhaitable que l'obtention d'une route optimale. On propose aussi la possibilité de scinder une grappe, de fusionner deux grappes ou d'échanger un membre entre deux grappes.

### 2.4.7 Gestion de logiciels et de services

D'autres recherches suggèrent l'utilisation des agents mobiles pour gérer et déployer des logiciels sur un réseau. Hall et al. (1999) ont décrit une architecture nommée « Software Dock ». Cette architecture intègre les agents mobiles pour permettre une décentralisation et pour faciliter l'analyse des composantes logicielles installées chez un client par rapport à celles disponibles chez un fournisseur. L'intelligence des agents mobiles pour gérer le déploiement des logiciels permet donc de faciliter la gestion des logiciels dans toutes les étapes de leur cycle de vie que ce soit l'installation, la mise à jour, la configuration ou la désinstallation.

## 2.5 Gestion de performance

La gestion de performance consiste essentiellement à prendre des mesures sur le réseau. Par une analyse de ces mesures, on peut en déduire la performance du réseau. La surveillance est une des principales tâches des agents mobiles dans les systèmes de gestion. Dans cette section, nous allons d'abord étudier le concept de surveillance de réseau basée sur les agents mobiles. Ensuite, nous aborderons les architectures

permettant l'équilibrage de la charge et la gestion de la qualité de service. Finalement, nous verrons une application des agents mobiles pour réduire le trafic de signalisation et optimiser la prise de mesures de performance dans un réseau cellulaire.

### 2.5.1 Surveillance et analyse de performance

La surveillance (monitoring) d'un réseau est utile à toutes les disciplines de gestion : la gestion de configuration, de sécurité, de fautes, de performance et de comptabilité. En effet, la surveillance permet de se renseigner dynamiquement sur l'état du réseau aussi bien à l'échelle locale que globale. Elle peut fort bien servir à détecter des fautes, des intrus ou faire le décompte des ressources à facturer.

La surveillance est la tâche la plus typique à confier aux agents mobiles. Elle consiste à envoyer des agents mobiles auprès d'un élément que l'on veut surveiller, en ayant soin de définir les paramètres qui devront être surveillés et la manière dont l'agent doit accumuler ou réagir à cette information. Cet agent ne peut activer une communication qu'en cas d'erreur ou de dépassement d'une valeur limite ou encore envoyer les informations dans une forme réduite, compressée ou filtrée (Gavalas et al., 2000). Plusieurs articles (Bieszczad et al., 1998; Liotta et al., 2002) parlent de surveillance sans toutefois en faire l'objet central des travaux. Pour ces raisons, nous n'avons sélectionné que les recherches donnant des applications spécifiques à la surveillance.

#### *Surveillance de services*

Les agents mobiles peuvent être employés pour surveiller la qualité d'un service donné à plusieurs usagers. Par exemple, Knight et Hazemi (1999) ont proposé l'utilisation des agents pour vérifier la qualité d'un service Web. Avec les agents mobiles, il est possible de vérifier la qualité de ce service sur le système de l'utilisateur. On obtient donc une vue plus claire de la situation, en raison de la localité, que si on ne se fiait qu'aux informations recueillies sur le côté serveur. Knight et Hazemi (1999) proposent l'utilisation de deux types d'agents : l'accumulateur et le collecteur. Le

premier migre vers une station client et lit diverses valeurs au choix du gestionnaire. Dans le cas du service Web, ces valeurs peuvent être le nombre de kilooctets pour les transferts et le délai. À intervalle régulier, il effectue un calcul d'agrégations sur les données. Le second agent se promène de station en station pour collecter les informations recueillies par les accumulateurs.

Günter et Braun (2002) ont proposé un système dont la motivation est de donner aux clients de services Internet un moyen d'obtenir des statistiques valables sur la qualité de leur service. Cette possibilité doit permettre au client un maximum d'informations sans toutefois compromettre la sécurité du réseau. Günter et Braun (2002) indiquent que les méthodes couramment utilisées ne sont pas très efficaces : soit qu'elles donnent trop peu de données (rapports), soit qu'elles ne sont pas en temps réel (analyse de trace) ou encore qu'elles causent des problèmes de sécurité (accès direct à SNMP). De plus, ils expliquent que l'utilisation de SNMP peut influencer les mesures. Ils ont donc eu l'idée d'introduire des agents mobiles à leur système. Un agent mobile peut s'exécuter sur un nœud CSM (Customer-based IP service monitoring) et interagir avec l'équipement ou un service via un « T-component ». Ce « T-component » est une interface entre l'agent et l'équipement. Ce dernier a pour but de copier rapidement les paquets entrants dans l'équipement pour permettre une analyse sans trop modifier le fonctionnement du réseau. Le CSM offre un certain nombre de services. L'agent est isolé dans le CSM pour assurer un maximum de sécurité et il ne peut voir que les paquets qui répondent aux exigences de deux filtres. Un premier filtre empêche l'accès à des paquets non autorisés tandis que le second permet de conserver les paquets choisis par le client. Les agents peuvent, selon l'intérêt du client, surveiller différents paramètres de performance.

***Surveillance d'un système en grille***

La surveillance d'un système en grille est abordée par Tomarchio et Vita (2001). Dans ce contexte, une grille est un ensemble d'ordinateurs reliés par un réseau à grande vitesse pouvant être utilisé pour accomplir un but ou une tâche en

parallèle. Cette grille n'est pas nécessairement composée d'équipements homogènes et n'a pas de structure définie. Les outils de gestion disponibles sont multiples et leur utilisation peut mener à des erreurs. Ces systèmes doivent être surveillés pour permettre des analyses de performance, la détection des erreurs et leur mise au point. Tomarchio et Vita (2001) indiquent que les agents mobiles sont parfaitement adaptés pour automatiser et effectuer les tâches de gestion d'une grille et proposent un système utilisant la plate-forme MAP. Les agents mobiles chargés de surveiller les nœuds de la grille minimisent le trafic causant moins de délais de communications. Cette réduction de trafic, considérant que les environnements parallèles sont souvent ralentis par les délais de communications, peut améliorer les performances des ordinateurs de la grille.

### 2.5.2 Performance des systèmes d'agents mobiles

Plusieurs recherches ont été effectuées pour démontrer formellement les avantages des agents mobiles en terme de performance. Nous avons déjà indiqué précédemment que l'utilisation des agents mobiles pour la gestion de réseaux devait faire l'objet de décisions intelligentes quant à leur dispersion, leur nombre et leur constitution. Nous présentons ici deux études qui ont évalué les performances d'un système d'agents mobiles.

Rubinstein et al. (2000, 2002) ont démontré qu'un système de gestion à base d'agents mobiles n'est pas toujours plus performant qu'une architecture classique SNMP et regardent si cela demeure vrai avec plusieurs agents. Pour faire cette démonstration, ils proposent un système simple où N agents mobiles se séparent le réseau en N partitions. Le système utilise les agents mobiles pour accéder à une base MIB-II sur un certain nombre de nœuds. Cette expérience a été faite sur deux ordinateurs qui s'échangeaient l'agent mobile de gestion à tour de rôle autant de fois que l'on voulait d'éléments à visiter. Cette expérience est soutenue par une simulation sur une topologie de type GT-ITM (Georgia Tech Internetwork Topology Models) dans NS (Network Simulator). On compare donc la méthode avec gestionnaire central utilisant SNMP et celle avec un agent mobile. Cet agent se déplace en communiquant

via un agent statique chargé de faire le pont entre l'agent mobile et l'agent SNMP. Les métriques observées sont le temps de réponse et l'utilisation du réseau en kilooctets.

Dans l'expérience, le temps de réponse est nettement plus long pour l'approche à un seul agent, et ce, peu importe le nombre d'éléments à gérer (de 1 à 240 dans l'étude). Pour ce qui est de l'utilisation du réseau en kilooctets, il faut attendre qu'il y ait plus de 240 éléments pour que l'approche à un seul agent devienne meilleure. L'expérience est cependant très limitée et n'offre pas une topologie réaliste.

Dans la simulation, les performances de chaque approche sont moins bien définies. Rubinstein et al. (2000, 2002) introduisent des mesures impliquant un, deux, quatre et huit agents mobiles travaillant en parallèle dans le réseau. Ils notent que la grandeur de la tâche de gestion influence beaucoup le trafic en kilooctets et le temps de réponse des deux méthodes. Une augmentation de la tâche de gestion favorise les agents mobiles pour l'utilisation du réseau, mais augmente aussi leur temps de réponse par rapport à la gestion à distance. La modification de la grandeur de l'agent mobile modifie aussi grandement l'utilisation du réseau. L'utilisation de plusieurs agents tend à réduire le temps de réponse. Cela se fait aux dépends du trafic qui devient plus grand dans le réseau. Selon le nombre de nœuds à visiter, le temps de réponse augmente linéairement avec la gestion à distance alors qu'il augmente exponentiellement avec les agents mobiles. Cette augmentation rapide est due à l'accroissement de la taille de l'agent mobile pendant son voyage. Les agents mobiles ont cependant un avantage, leur performance ne change pas en fonction de la topologie, comme c'est le cas de la gestion à distance. Ce fait peut être attribué à la surutilisation de certains liens dans une topologie réelle lorsque SNMP est utilisé à distance.

La simulation révèle aussi un fait intéressant en comparant l'utilisation de TCP (Transmission Control Protocol) et de UDP (User Datagram Protocol) comme moyens de communications et de transport. Le temps de réponse des agents mobiles est diminué grandement par l'utilisation de UDP alors que cela n'a aucun effet perceptible sur l'approche à distance. Rubinstein et al. (2000, 2002) attribuent ce fait au mécanisme de départ lent de TCP et à la faible taille des paquets générés par SNMP.

En conclusion, Rubinstein et al. (2000, 2002) indiquent que l'utilisation d'un système de gestion à base d'agents mobiles pour des fins de performance peut être bénéfique si le nombre d'éléments à gérer se situe entre deux valeurs. Ces deux valeurs dépendent du nombre d'agents utilisés, de leur grandeur initiale et de la grandeur moyenne de la tâche à accomplir. Il faut donc être conscient que l'ajout d'agents mobiles pour la gestion doit se faire de façon éclairée. Bien entendu, la performance n'est pas le seul critère pour intégrer les agents mobiles à la gestion. Elle n'est même pas toujours un critère pour ce choix.

### 2.5.3 Équilibrage de charge

Beaucoup de recherches tentent d'utiliser les agents mobiles pour permettre d'équilibrer la charge d'un réseau, d'une composante ou d'un service. L'équilibrage de charge se fait en coopération avec la surveillance du réseau.

Gavalas et al. (2002) ont affirmé qu'il n'est pas efficace de diviser le réseau en $\underline{N}$ partitions pour $\underline{N}$ agents mobiles, étant donné que cette division ne limite pas le nombre de transferts d'agents mobiles. Ils proposent une solution dynamique où les gestionnaires sont eux-mêmes des agents mobiles qui peuvent se déplacer dans le réseau. Ces gestionnaires sont appelés MDM (Mobile Distributed Manager). Un gestionnaire mobile est responsable du domaine qui lui est attribué. Si la charge augmente sur la machine où il est installé, il a la possibilité de se déplacer. D'un autre côté, si le nombre d'éléments à gérer augmente, le MDM peut créer et assigner un autre MDM pour continuer une gestion optimale du réseau. Cette distribution de l'intelligence de gestion permet la continuité de la gestion du réseau en cas de partitions ou d'erreurs. Un MDM peut gérer les éléments de son domaine avec les techniques de gestion de son choix.

Liotta et al. (2002) ont proposé un système similaire servant à surveiller le réseau et à adapter le positionnement des stations de gestion dynamiquement en fonction de l'état du réseau. Ils ont introduit le concept de clonage et ne se limite, contrairement à Gavalas et al. (2002), qu'à la surveillance. Une station voulant

surveiller le réseau lance un ou plusieurs agents mobiles dans des partitions logiques initiales. Ces agents mobiles évaluent à chaque saut s'il se trouve dans une position optimale pour surveiller le réseau. Cette évaluation se fait par l'agent à l'aide des tables de routage sur les nœuds visités. Les agents mobiles se clonent pour permettre une dispersion efficace sur le réseau. Ce clonage permet de sauver de la bande passante, en lançant les agents mobiles localement, sans réutiliser le chemin entre la station d'origine et le nœud local. Les agents de surveillance réévaluent périodiquement le coût associé à une relocalisation. S'ils trouvent un meilleur emplacement, ils migrent. Les résultats obtenus montrent clairement que cet algorithme réagit très bien aux changements tels que des bris de liens. Lorsque le ratio du nombre d'agents mobiles au nombre d'éléments à gérer augmente au-dessus de 0.3, la surveillance n'augmente que légèrement le temps de réponse moyen et le trafic moyen en cas de bris de liens. Selon les expériences, ces valeurs ont presque doublé dans le scénario sans agents mobiles.

Kim et al. (2001) ont proposé d'utiliser les agents mobiles pour permettre l'équilibrage de la charge dans un serveur Web. Les agents mobiles sont responsables d'analyser les journaux des différents serveurs Web pour déterminer s'ils sont surutilisés. Les agents mobiles vérifient différents paramètres et commandent une dispersion. Dans une autre étude de Suri et al. (2001), on propose d'utiliser des agents mobiles pour transporter des calculs sur des stations de travail inutilisées ou libres. L'agent mobile peut s'installer sur une machine et repartir lorsque la station est réutilisée. Ce système utilise une mobilité forte, ce qui implique que l'agent mobile peut reprendre son exécution là où il l'avait laissée avant de quitter une machine.

### 2.5.4 Gestion de qualité de service (QoS)

La Corte et al. (2001) ont proposé de complémenter RSVP (Reservation Protocol) en introduisant des agents mobiles pour gérer les portions du réseau contenant des nœuds n'utilisant pas RSVP. RSVP est un protocole permettant la réservation de ressources et la gestion de qualité de service. Il ne peut respecter les contraintes de

qualité de service que si toute la route contient des nœuds RSVP. Dans le cas contraire, les nœuds qui n'ont pas RSVP sont ignorés et cela implique donc que les contraintes de qualité de service peuvent ne pas être respectées. La Corte et al. (2001) décrivent les nœuds n'implémentant pas RSVP comme des nuages. Une route choisie dans ce nuage s'appelle un tunnel. Le but de leur solution est d'utiliser des agents mobiles pour surveiller les rebords de différents tunnels potentiels. Les agents vérifient si les contraintes de qualité sont ou peuvent être respectées. Ces agents peuvent facilement relocaliser la connexion d'un tunnel à un autre en cas de bris de lien ou de dégradation de performance.

Dans les travaux de Manvi et Venkataram (2000), les agents mobiles assurent un contrôle de bout en bout de la qualité de service d'une connexion. L'architecture utilise plusieurs types d'agents statiques et mobiles. Les agents mobiles s'acquittent de la négociation et de la renégociation de qualité de service ainsi que de la surveillance de la performance de la connexion. Le premier agent mobile voyage de nœuds en nœuds en négociant les paramètres de qualité de service avec le négociateur local et en réservant temporairement les ressources. S'il peut obtenir la qualité de service désirée et les ressources nécessaires sur toute la route, l'agent mobile négociateur revient sur ses pas pour établir la connexion en concordance avec les paramètres négociés et en réservant les ressources de façon permanente. Les agents mobiles de surveillance se promènent sur le réseau en vérifiant que les critères de performance de la connexion sont respectés. Si ce n'est pas le cas, ces agents de surveillance avertissent un agent statique situé sur la station du demandeur du service. Ce dernier vérifie s'il peut pallier ce problème localement. Si ce n'est pas le cas, il envoie un agent mobile « renégociateur » qui doit tenter de retrouver les mêmes paramètres de qualité sur une autre route.

### 2.5.5 Performance des réseaux de télécommunications et cellulaires

Brusic et al. (2000) ont démontré que les éléments actuels du réseau téléphonique cellulaire n'auront pas assez de capacité pour contenir la montée du trafic.

Le but de cette étude est de diminuer le trafic de signalisation en employant des agents mobiles. L'agent mobile est déployé sur le cellulaire pour obtenir des informations sur le profil d'utilisation de l'usager. Lorsque ce profil est connu, il est utilisé pour réduire le trafic de signalisation en employant un modèle probabiliste d'utilisation. Pour un usager urbain utilisant toujours le même profil, la signalisation diminue. Dans le cas d'un usager moins « stable », elle demeure la même.

Brusic et al. (2000) introduisent aussi une autre idée pour détecter les zones non couvertes. Ils indiquent qu'un agent mobile peut facilement effectuer des mesures sur les zones non couvertes et ainsi éviter d'envoyer un camion contenant de l'équipement spécialisé pour effectuer les mesures. L'agent mobile effectue sa tâche pendant la période déconnectée et transmet ses mesures lors du retour dans une zone couverte.

## 2.6 Gestion de fautes

La gestion de fautes est un domaine plus complexe, étant donné que le gestionnaire doit pouvoir réagir à des conditions d'opérations souvent non prévues. Dans la gestion de fautes, il y a des approches proactives et réactives. Les approches proactives ont certains avantages surtout dans le domaine des télécommunications où le nombre de pannes doit être minimisé. Cependant, les techniques de gestion de fautes proactives introduisent plusieurs problèmes d'implémentation ainsi que les inévitables fausses alarmes. Elles sont, pour l'instant, plus l'apanage des systèmes à agents intelligents et statiques. Les principales recherches avec des agents mobiles présentent des techniques réactives permettant au minimum de localiser les fautes. Certaines autres, plus ambitieuses, visent le diagnostic et même la réparation des fautes.

### 2.6.1 Modèle d'un système de gestion de fautes

Le but d'utiliser les agents mobiles dans la gestion de réseaux est d'automatiser un maximum de tâches. El-Darieby et Bieszczad (1999) ont proposé une architecture où le pouvoir de gérer les fautes est délégué aux agents mobiles. Le réseau est séparé en plusieurs partitions logiques. Chacune de ces partitions est contrôlée par un sous-

gestionnaire. Lorsque ce dernier découvre un problème dans sa section, il le signale au gestionnaire central et il reçoit la visite d'un agent mobile transportant l'intelligence requise pour la gestion de fautes. L'agent mobile fait une enquête sur les sous-gestionnaires en lisant les messages et les alarmes causés dans ce domaine. Il peut ensuite se définir un itinéraire pour vérifier auprès des éléments du réseau quelle est la véritable cause et quelle pourrait être la solution au problème. Le but avoué de cette étude est de diminuer le temps qu'un système est en panne et minimiser le recours aux opérateurs humains.

### 2.6.2 Localisation de fautes selon le principe de l'essaim

Le principe de l'essaim est pris des colonies de fourmis. White et Pagurek (1999) et White et al. (1998) ont proposé un système où l'on utilise des agents mobiles simples et légers pour accomplir une tâche de groupe. Le but est de réduire les transmissions au gestionnaire d'alarmes causées par des erreurs du réseau. Ils utilisent des agents possédant une fonction de migration, une faible mémoire et la possibilité d'émettre et d'interagir avec des traces « chimiques ». Le principe est simple. Deux principaux types d'agents mobiles se promènent à l'intérieur du réseau. Le premier type d'agent est inspiré du système immunitaire. Les agents de ce type peuvent être spécialisés selon les différents types d'équipements et de manufacturiers du réseau. Lorsqu'ils perçoivent un problème, ils laissent des traces sur leur passage. Ces problèmes sont généralement des problèmes de performance. Le second type d'agent se promène continuellement à l'intérieur du réseau pour repérer les endroits où la concentration de messages « chimiques » est la plus grande. Plusieurs autres agents permettent l'évaporation des messages et la découverte de liens servant à l'exploration continuelle. La localisation de la faute devient donc l'endroit contenant le plus de traces chimiques. Étant le nœud le plus visité, selon le principe des colonies de fourmis, il devient la localisation la plus probable.

### 2.6.3 Localisation de fautes par corrélation d'alarmes

Les travaux de Lipperts (1999) traitent du problème inhérent au trop grand nombre d'alarmes générées par les réseaux actuels. Les alarmes sont des messages évoquant des problèmes de performance ou de sécurité, ou encore signifiant des erreurs. Selon Lipperts (1999), ces alarmes sont trop nombreuses et trop peu localisées pour permettre à un opérateur humain de les utiliser. Lipperts propose donc d'améliorer les principes de corrélation d'alarmes en utilisant les agents mobiles. La corrélation d'alarmes consiste à traiter les alarmes reçues de façon à réduire l'information et à fournir de l'information plus claire et plus précise. Selon l'état du réseau, il propose de lancer des agents mobiles nommés observateurs, corrélateurs et agents de topologie. Les premiers s'installent sur un élément du réseau et s'enregistrent comme destinataires pour les alarmes. Ils peuvent faire un prétraitement de l'information. Les corrélateurs implémentent un algorithme de corrélation au choix et visitent chaque observateur pour effectuer une corrélation globale des alarmes. Finalement, les agents de topologies sont chargés de maintenir une image des composantes actives du réseau. La corrélation de plusieurs alarmes devrait donc permettre de mieux localiser la vraie source d'un problème.

### 2.6.4 Localisation de fautes par analyse du trafic du réseau

Zhang et Sun (2001) ont utilisé des agents mobiles intelligents pour détecter des erreurs dans un réseau IP (Internet Protocol). L'« outil » mis à la disposition des agents mobiles est ICMP (Internet Control Message Protocol). Cet outil ne permet pas d'obtenir des informations utiles pour un réseau modérément gros et complexe. Ici, Zhang et Sun (2001) mettent à profit la mobilité des agents pour pallier ce problème. Leur architecture installe une plate-forme d'exécution sur chaque élément du réseau. Lors d'une détection d'un hôte qui n'est plus atteignable par un agent statique dans le gestionnaire, plusieurs agents mobiles intelligents sont lancés pour enquêter. Ils enquêtent en surveillant les paquets de contrôle ICMP localement et isolent les symptômes en analysant ces paquets de contrôle. Les agents mobiles sur chaque

élément communiquent entre eux pour isoler la cause du problème. L'analyse du trafic permet donc la localisation de la faute.

### 2.6.5 Surveillance de santé du réseau

Nous avons déjà vu la surveillance de réseau dans la section sur la gestion de performance. La surveillance peut aussi servir la cause de la gestion de fautes en tentant de réagir avant qu'une défaillance ait lieu. Un agent mobile peut être dédié à surveiller certaines valeurs. Si ces valeurs sont dépassées ou deviennent anormales (analyse du trafic par exemple), l'agent mobile peut en informer le gestionnaire. Cette idée vient du constat que certaines pannes produisent préalablement une baisse de performance et laissent entrevoir des signes avant-coureurs.

### 2.6.6 Tolérance aux fautes d'un système à base d'agents mobiles

Un système de gestion à base d'agents mobiles devrait pouvoir gérer ses propres fautes et être tolérant aux fautes. Par exemple, si un agent mobile s'exécute sur un nœud qui cesse de fonctionner, il faut pouvoir s'en rendre compte et récupérer l'exécution de l'agent. Cette tolérance aux fautes se fait généralement par redondance et surveillance des agents mobiles. Elle est essentielle s'il est envisagé d'utiliser uniquement les agents mobiles pour la gestion de réseaux. Cette gestion peut être paralysée si le système d'agents mobiles n'est pas robuste.

## 2.7 Gestion de sécurité

La gestion de sécurité est un domaine aussi important que les autres domaines abordés jusqu'ici. Certaines organisations négligent cet aspect de la gestion de réseaux et risquent d'en faire leurs frais tôt ou tard (Huston, 1999). Certaines études indiquent que les tentatives d'accès non autorisées se multiplient d'année en année (Bernardes et Dos Santos Moreira, 2000). Il est donc important de se protéger contre ces attaques et d'améliorer les techniques pour les contrer ou du moins les détecter.

Les agents mobiles offrent des perspectives intéressantes pour la gestion de sécurité. Cependant, les systèmes à base d'agents mobiles sont sujets à des problèmes de sécurité qui leurs sont propres. On peut donc comprendre que leur introduction ne soit pas encore acceptée dans ce domaine de gestion. On peut facilement envisager qu'un intrus puisse tromper un tel système en injectant des agents mobiles hostiles ou en modifiant des agents mobiles appartenant au réseau. Cependant, beaucoup d'efforts ont été déployés pour contrer ces problèmes de sécurité.

### 2.7.1 Résistance aux attaques de déni de service

Selon Mell et al. (2000), aucun hôte ne peut être complètement immunisé contre les attaques de déni de service (Denial of Service). Il est donc possible pour un intrus de désactiver un système de détection d'intrusion en l'inondant de paquets de données inutiles. Selon eux, la meilleure protection contre une telle tentative est de relocaliser les composantes servant à la détection d'intrus vers des hôtes qui ne sont pas attaqués. Ces composantes, implémentées à l'aide d'agents mobiles, peuvent ainsi être cachées aux intrus dans le réseau, rendant pratiquement impossible la désactivation du système de sécurité.

On peut noter que cette idée peut aussi s'avérer utile pour relocaliser des services importants non reliés à la sécurité.

### 2.7.2 Détection des intrusions

La détection d'intrusion bénéficie d'un des avantages des agents mobiles plus que toute autre discipline. Leur répartition et leur possibilité de fonctionner sans contrôle centralisé permettent de créer des systèmes de détection ne pouvant pas être mis hors service facilement (Mell et McLarnon, 2002) ou encore être corrompus (Bernardes et Dos Santos Moreira, 2000) et chaque entité peut être répliquée, remplacée et sacrifiée (Foukia et al., 2001). De plus, selon Bernardes et Dos Santos Moreira (2000), l'aspect statique des approches classiques demande une reconstruction complète chaque fois qu'une nouvelle forme d'intrusion fait surface. Toutes ces

indications relatant de l'adaptabilité des agents mobiles plaident en faveur de leur utilisation dans la détection d'intrusion.

Selon Krügel et al. (2001), les approches de détections classiques telles que le pare-feu installé à un point précis du réseau ne peut détecter toutes les intrusions. Le travail ravageur d'un vers qui effectue son travail à l'intérieur du réseau ou encore d'une chaîne de connexions « Telnet » ne peut être détecté sur un simple point du réseau. Ces problèmes de sécurité deviennent évidents en regardant l'ensemble du réseau (Krügel et al., 2001). Ils soutiennent qu'un système à base d'agents distribués est plus efficace, et que les agents mobiles peuvent faire plus efficacement ce travail. De plus, selon Bernardes et Dos Santos Moreira (2000), 70 % des attaques proviennent de l'intérieur de l'organisation, rendant inutiles les approches impliquant un système de détection à un seul emplacement.

Les agents mobiles servent la détection d'intrus à différents niveaux. Ils peuvent tout simplement être utilisés pour optimiser l'analyse des historiques sur chaque nœud. Ils sont aussi utilisés comme dépisteurs pour retrouver l'origine d'une intrusion, comme enquêteurs pour trouver une série d'évènements potentiellement malicieux ou encore dans le cadre d'un système calquant le système de défense du corps humain. Plusieurs autres variantes existent, dont des approches qui combinent plusieurs de ces idées.

***Dépistage des intrusions***

Asaka et al. (1999), on propose un système simple où les agents mobiles ont comme rôle de dépister l'origine des intrusions et de produire des rapports sur le type d'attaque. Asaka et al. (1999) identifient quatre phases à une intrusion. Les intrus recherchent d'abord les vulnérabilités du réseau. Ils entrent ensuite dans une phase d'activité, où ils pénètrent dans le réseau. Ensuite, ils exécutent leurs actions. Cette phase laisse des marques visibles. Finalement, ils effacent leurs traces pendant la phase de mascarade. Asaka et al. (1999) proposent donc de surveiller les traces pendant l'action des intrus à l'aide de senseurs statiques. Certaines traces, comme un

changement de mot de passe, un accès « root » par un processus non « root » ou une modification d'un fichier sensible peuvent signaler une intrusion potentielle (Asaka et al., 2002). On lance alors un agent mobile dépisteur qui tente de retrouver la trace d'une intrusion. Cet agent mobile active plusieurs agents statiques sur les nœuds qu'il visite. Ces agents statiques collectent et enquêtent pour obtenir plus d'informations sur l'attaque et fournir un rapport de l'invasion.

***Corrélation d'évènements (SPARTA)***

SPARTA (Security Policy Adaptation Reinforced Through Agents) de Krügel et al. (2001) est une plate-forme de détection d'intrusions qui propose d'utiliser les agents mobiles pour corréler des évènements distribués. Krügel et al. (2001) indiquent qu'un événement à lui seul ne peut conclure à une intrusion. C'est l'identification d'un motif d'évènements qui peut mener à cette conclusion. Ils ont élaboré un langage nommé EQL (Event Query Language) qui permet de rechercher des ensembles d'évènements. Par exemple, un évènement pourrait être une tentative d'accès qui a échoué. L'architecture de SPARTA comprend un générateur local d'évènements, une base de données d'évènements locale et une plate-forme pouvant exécuter des agents mobiles. Les agents mobiles, lancés par des opérateurs du réseau, sont chargés de détecter des motifs décrits avec EQL. Il n'y a donc aucune entité centrale pour faire la corrélation des évènements. Quand un agent est lancé, il contacte d'abord un répertoire qui l'aide à choisir une route en fonction de sa tâche. Ensuite, il migre de nœuds en nœuds en recherchant le motif qui lui a été assigné.

D'autres auteurs proposent des approches semblables à SPARTA. Helmer et al. (2002) ont proposé un système à trois couches. La couche du bas sert à collecter des informations utiles sur chaque nœud, comme les historiques d'un système ou d'une application. La couche du haut sert à organiser la connaissance, ainsi que les communications des agents mobiles. La couche du milieu est celle des agents mobiles, qui ont pour tâche de vérifier l'information récente obtenue sur chaque nœud et, par l'entremise de communication, en vient à déduire s'il y a une intrusion ou non.

Bernardes et Dos Santos Moreira (2000) ont présenté un système en quatre couches où seule la coopération entre agents peut mener à la conclusion d'une intrusion. Ces systèmes, bien que différents architecturalement, accomplissent leur but par la coopération de différents agents statiques et mobiles.

***Système inspiré du système immunitaire humain***

D'autres approches sont basées sur le système immunitaire humain. Dans ces systèmes, les agents mobiles jouent le rôle d'entités simples mais efficaces. Foukia et al. (2001) ont présenté une idée où les systèmes d'ordinateurs se défendent comme le corps humain. Le système proposé installe temporairement un agent mobile sur une composante du réseau. Après un certain temps, cet agent migre vers un autre nœud, soit de manière aléatoire, soit selon une fonction statistique. Selon Foukia et al. (2001), cette idée permet de réduire la charge sur le réseau en évitant une surveillance constante, tout en assurant une bonne couverture. Plusieurs agents mobiles sont présents en même temps dans le système avec chacun des fonctions et des tâches spécifiques. Cette spécialisation est à l'image du système immunitaire, qui comporte des défenses différentes pour différents maux.

Dans les travaux de Dasgupta et Brian (2001), on définit un système similaire appelé SANTA (Security Agents for Network Traffic Analysis). Ce système utilise plusieurs agents mobiles. Des agents mobiles de surveillance sont chargés de vérifier les opérations qui sortent de l'ordinaire. Ils peuvent ensuite communiquer de telles opérations à des agents mobiles de décisions, via des agents de communication. Par logique floue, les agents de décisions déterminent l'importance de l'anomalie et décident d'une action à entreprendre. Une action peut être de notifier un opérateur humain ou de lancer un agent mobile tueur pour terminer les processus représentant les plus hautes sources d'anomalies. Le système ne peut jamais être sûr d'être en présence d'une véritable intrusion et peut introduire de fausses alertes.

Les deux systèmes doivent d'abord subir une phase d'apprentissage. Ils peuvent ensuite servir à détecter des anomalies. La phase d'apprentissage permet

d'obtenir des informations sur l'utilisation normale du réseau pour ensuite permettre aux divers agents mobiles composant ces systèmes de découvrir les intrusions.

### 2.7.3 Mini pare-feu et réaction aux intrusions

Hwang et Gangadharan (2001) ont exploré la possibilité d'utiliser des agents mobiles pour mettre à jour dynamiquement les politiques de sécurité des pare-feu. Ils utilisent une architecture composée d'un pare-feu de passerelle, un gestionnaire de politique de sécurité et un mini pare-feu installé sur chaque nœud du réseau intranet. Des senseurs permettent de détecter une intrusion. Un agent mobile, doté d'une certaine intelligence, peut mettre à jour, en temps réel, toutes les politiques de tous les pare-feu lors d'une intrusion. Ces minis pare-feu constituent une seconde ligne de défense et peuvent réagir en temps réel grâce aux agents mobiles intelligents. Ainsi, le réseau peut s'adapter rapidement à une intrusion et y résister avec un minimum de dommages.

## 2.8 Gestion de comptabilité

La gestion de comptabilité est de toute évidence le domaine de gestion le plus négligé. Certaines applications aux agents mobiles vues au sein des autres domaines de gestion touchent aussi directement ou indirectement à la gestion de comptabilité. Ainsi, la détection d'intrusions peut aussi servir à détecter la mauvaise utilisation des ressources (Asaka et al., 1999). La gestion de comptabilité doit aussi garder des statistiques sur l'utilisation des ressources. Cela peut être fait par les systèmes de surveillance à base d'agents mobiles. Il n'y a cependant pas de recherches notables effectuées pour utiliser les agents mobiles à des fins de gestion de comptabilité.

# CHAPITRE 3

# ARCHITECTURE DE SYSTÈME DE GESTION PROPOSÉE

Il est maintenant évident que les réseaux de télécommunications actuels ne sont plus des systèmes disposant d'entités homogènes d'un seul manufacturier et utilisant un seul protocole. En réalité, un réseau est composé de plusieurs entités différentes obligeant l'utilisation de plusieurs protocoles et interfaces de gestion. Les différents besoins des clients et la croissance des utilisateurs exigent l'adoption de plusieurs technologies différentes. Les utilisateurs peuvent nécessiter de nouveaux moyens d'accès, d'une meilleure qualité de service ou d'un accès plus rapide. De leur côté, les administrateurs peuvent varier les technologies en fonction de leur coût, de leur disponibilité et de leur performance. L'architecture proposée prend donc en compte cette hétérogénéité inévitable des réseaux. Elle vise à utiliser les agents mobiles pour effectuer et automatiser les tâches de gestion dans un réseau de n'importe quelle grandeur et englobant plusieurs technologies.

Tout d'abord, dans ce chapitre, nous spécifierons les requis de conception de ce système de gestion en relation avec les problématiques des réseaux hétérogènes actuels. Ensuite, nous présenterons l'architecture générale et ses composantes. Nous enchaînerons avec la caractérisation du réseau de gestion et du réseau à gérer propre à notre architecture. Nous terminons avec les modèles de déplacements et quelques algorithmes de gestion.

## 3.1 Requis de conception

La perpétuelle évolution des besoins et des réseaux d'aujourd'hui force les manufacturiers de composantes de réseaux à constamment améliorer leurs produits. La multitude de composantes et de protocoles ajoutés à cette évolution constante oblige ces manufacturiers à dépenser leurs efforts sur l'optimisation, la performance et les

innovations technologiques de leurs produits plutôt que sur les systèmes de gestion de ces derniers. Lorsque de tels systèmes émergent finalement, ils sont propriétaires, c'est-à-dire élaborés par le manufacturier et spécifiques à une série limitée de composantes. Ce problème est amplifié par le fait que les réseaux sont maintenant colossaux et que la gestion demande des ressources humaines en grande quantité. À cela, on ajoute un nombre toujours croissant d'alarmes relatant des problèmes parfois dépendants, parfois indépendants, à travers tout le réseau. Dans le contexte actuel, les administrateurs du réseau doivent utiliser plusieurs outils de gestion et il devient difficile de résoudre dans un temps record les problèmes les plus banals. Il faut aussi ajouter que, dans de vastes réseaux de télécommunications, l'administration de diverses sections du réseau relève de départements distincts. Dans cette optique, il n'est pas toujours simple d'effectuer des tâches de gestion diverses à la grandeur du réseau. Nous sommes donc encore loin d'une approche standardisée qui permettrait l'unification des systèmes de gestion en un seul. Il apparaît donc qu'un système qui aurait pour but de gérer l'ensemble d'un réseau hétérogène doit le faire en composant avec des outils de gestion tout aussi hétérogènes.

Le but premier de ce chapitre est de présenter une architecture pouvant s'intégrer rapidement et efficacement aux réseaux actuels. Il est impératif que l'architecture soit flexible pour faciliter son évolution et son amélioration. Ce sont les qualités uniques des agents mobiles qui justifient leur intégration dans cette architecture telles que leur mobilité, leurs performances, leur possibilité de répartir la charge et d'utiliser au minimum les ressources les plus limitées des réseaux actuels. Ils ont aussi l'avantage d'être plus évolutifs et ont la possibilité d'accéder à plusieurs ressources et technologies qui ne sont pas, à priori, disponibles à distance. L'architecture ne doit pas être invasive en permettant au réseau de fonctionner sans le système de gestion à base d'agents mobiles et en évitant une trop grande appropriation des ressources du réseau.

Les réseaux actuels posent un certain nombre de problèmes récurrents et exigent des tâches de gestion répétitives qui peuvent être effectuées avec des procédures bien définies. Bien que de telles procédures existent, elles demeurent difficiles à appliquer

pour un opérateur humain en raison de la grandeur et de l'hétérogénéité du réseau. De plus, le processus peut se révéler ardu par la multitude d'interfaces différentes que les opérateurs doivent maîtriser. Bien qu'ils en viennent à maîtriser un bon nombre d'interfaces, le processus peut demeurer lent et fastidieux. Les agents mobiles aident à gérer cette complexité et accélèrent l'exécution de ce processus.

Les réseaux actuels sont encore très peu automatisés. Ce manque d'automatisation empêche les opérateurs du réseau de se consacrer aux problèmes et aux tâches complexes en étant constamment obligés d'effectuer une multitude de tâches de gestion et de résoudre un nombre encore plus grand de problèmes. L'automatisation de la gestion du réseau pourrait se faire avec des procédures à distance ou selon le modèle client-serveur. Cependant, un bon nombre d'informations et de procédures ne sont pas directement et immédiatement disponibles à distance. Un gestionnaire centralisé ou des gestionnaires répartis sur différents points du réseau, qui effectuent toutes les tâches à distance, peuvent facilement être surchargé en plus de causer du trafic inutile sur le réseau. Il en ressort donc qu'un certain nombre de problèmes inhérents à la gestion de réseaux pourraient être solutionnés efficacement par des agents mobiles intelligents qui, en migrant de stations en stations, effectueraient des tâches de gestion. C'est en effet la conclusion de plusieurs travaux que nous avons vus dans le chapitre précédent. Cependant, l'utilisation d'agents mobiles pour résoudre des problèmes encore dédiés à des opérateurs humains cause aussi une multitude de problèmes.

Il n'existe pas encore, dans la littérature, de solutions de gestion à base d'agents mobiles pour des réseaux hétérogènes ayant des applications réelles et définitives dans l'industrie. Les raisons en sont multiples. Les agents mobiles, dans leur forme actuelle, sont encore très récents. Ils font l'objet de beaucoup d'ouvrages de recherche théoriques et de simulations. Ils n'ont donc pas tous été éprouvés dans un contexte d'utilisation réel. La seule utilisation d'agents mobiles ne garantit pas de meilleurs résultats que les solutions traditionnelles client-serveurs. L'architecture proposée a

comme requis de faire la démonstration de la pertinence des agents mobiles et de présenter des performances plus qu'acceptables.

Le mode d'interaction des agents mobiles avec leur environnement cause aussi un problème de taille. Un agent indépendant et pouvant prendre en compte différents scénarios a de fortes chances d'avoir une grande taille et un impact plus élevé sur le réseau. Un agent allégé doit être aidé par des interfaces uniformes, compliquant le déploiement de l'architecture et augmentant les communications entre agents. Le choix du mode d'interaction de l'agent mobile de gestion exige des compromis entre l'évolutivité, la facilité de déploiement de l'architecture, la performance et les fonctionnalités de l'agent. L'architecture de système de gestion doit donc utiliser un compromis avantageux entre les approches énoncées précédemment pour répondre aux requis de conception qui lui sont imposés.

Avec l'utilisation des agents mobiles, nos travaux doivent envisager une solution donnant la possibilité d'intégrer les technologies et les systèmes de gestion déjà en place. Il devient donc possible de poursuivre la gestion et l'opération du réseau sans les agents mobiles. Cependant, l'architecture a pour but de simplifier la gestion de ce réseau par l'utilisation des agents mobiles de gestion. Elle doit faciliter la mise en œuvre des tâches de gestion en fournissant des outils simples à utiliser.

En résumé, les agents mobiles de ce système de gestion visent à combler les lacunes existantes dans les grands réseaux de télécommunications tout en démontrant leur utilité dans un contexte d'utilisation différent et réel. Ils doivent simplifier la gestion du réseau en gérant les éléments complexes du réseau. Les agents mobiles doivent pouvoir utiliser efficacement les différentes technologies disponibles sur le réseau pour effectuer leurs tâches de gestion. L'architecture doit être souple, non invasive et facilement renouvelable.

## 3.2 Présentation générale de l'architecture et de ses composantes

Dans cette section, nous ferons une présentation générale des diverses composantes de l'architecture. Nous commencerons par définir quelques termes

utilisés pour décrire l'architecture. Ensuite, il sera question des différents concepts permettant de décrire l'architecture.

### 3.2.1 Définitions

Certains termes sont introduits et expliqués plus loin. Néanmoins, nous pouvons définir immédiatement certains termes utilisés dans ce chapitre pour permettre de bien comprendre comment ces derniers sont utilisés.

Une tâche de gestion est un terme générique désignant toute activité entrant dans le cadre de la classification ISO telle que la gestion de configuration, de performance, de comptabilité, de sécurité et de fautes.

Une station de gestion est un lieu permettant la gestion d'une ou de plusieurs composantes du réseau. Généralement, c'est un lieu où l'agent mobile pourra s'exécuter quand une composante ne peut pas le recevoir. Un ensemble de stations de gestion et de composantes pouvant accueillir des agents mobiles est appelé réseau de gestion. L'ensemble des composantes devant être gérées constitue le réseau à gérer. La nuance entre réseau de gestion et réseau à gérer est importante, car elle définit les limites du modèle de gestion. Par exemple, dans un réseau où toutes les composantes peuvent accueillir des agents mobiles et que toutes ces composantes doivent être gérées, on dirait que le réseau à gérer et le réseau de gestion sont identiques.

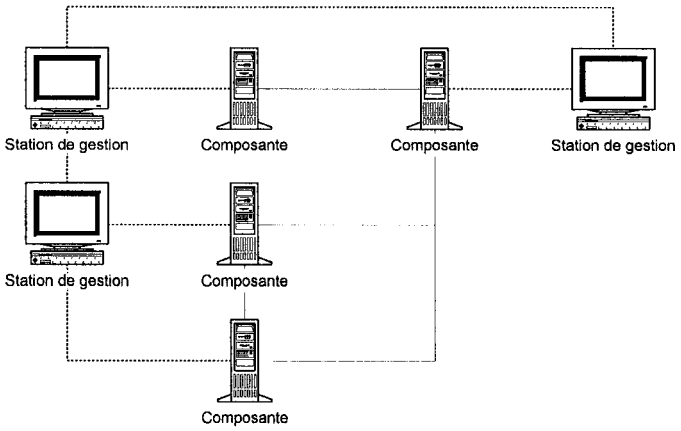


**Figure 3.1 Exemple de stations de gestion et composantes à gérer**

La Figure 3.1 montre la différence entre réseau à gérer et réseau de gestion. Le réseau à gérer est constitué de composantes et de liens en traits pleins. Le réseau de gestion est pour sa part composé des stations de gestion et de liens en traits pointillés. Les données utiles circulent sur les liens pleins, tandis que les données de contrôle (gestion) circulent sur les liens pointillés.

Le gestionnaire est un mot passe-partout indiquant une entité ou une personne qui fait autorité pour créer automatiquement ou sur demande des agents mobiles implémentant des tâches de gestion.

L'approche client-serveur optimisée est un modèle où les requêtes à distance sont minimisées et optimisées pour réduire au maximum le trafic et le temps de réponse. Généralement, pour éviter une concurrence déloyale, l'approche avec agents mobiles doit être comparée en terme de performance avec l'approche client-serveur optimisée. Cependant, plusieurs technologies utilisées aujourd'hui sont encore peu optimisées.

### 3.2.2 Intégration des agents mobiles

Nous savons, suite à divers travaux (Rubinstein et al., 2002; Rubinstein et al., 2000), que la performance de l'approche client-serveur n'est plus aussi bonne lorsqu'il est question de la comparer avec les agents mobiles sur des topologies non structurées comme celle d'Internet.

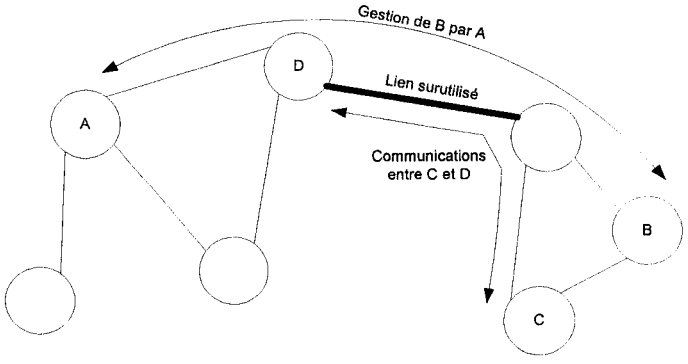


**Figure 3.2 Exemple d'un lien sollicité inutilement par un gestionnaire distant**

S'il faut gérer une multitude de composantes qui sont disposées dans différents sous-réseaux séparés par des liens, l'avantage des agents mobiles se fait fortement sentir. Ils n'occupent jamais en permanence un ou plusieurs liens et empêchent ainsi des problèmes de congestion qui ne se manifestent pas sur des topologies équilibrées. À la Figure 3.2, nous remarquons qu'avec une gestion à distance typique, si la station A doit gérer la station B, alors elle utilise de façon régulière le même lien qui est utilisé par la station C pour communiquer avec la station D. Nous croyons donc que les agents mobiles sont un vecteur avantageux pour la gestion de réseaux à grande échelle. L'approche client-serveur a le désavantage de devoir dédier une ou plusieurs stations qui exécutent l'ensemble des tâches de gestion. Les agents mobiles offrent un traitement beaucoup plus parallèle (Figure 3.3). Il est donc moins courant que certaines tâches de gestion retardent sur une station de gestion trop sollicitée. Ils mettent donc à profit, dans notre architecture, leur principal avantage qu'est la mobilité.

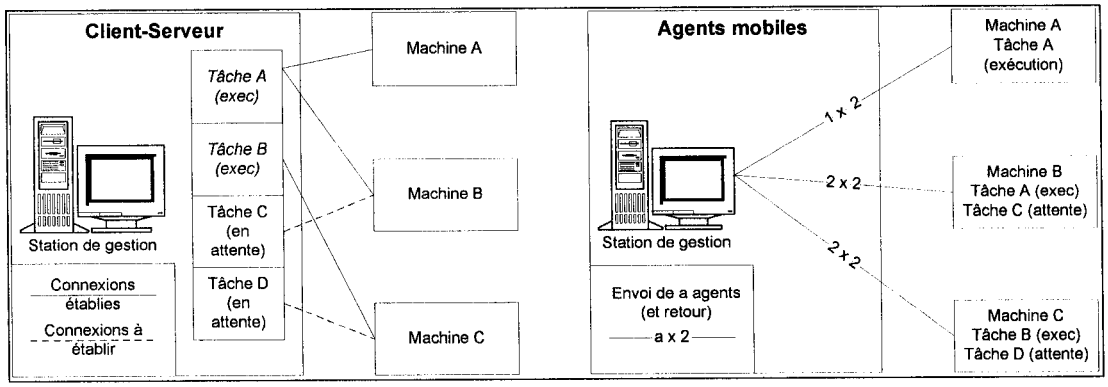


**Figure 3.3 Exemple du parallélisme des agents mobiles**

À la Figure 3.3, le gestionnaire du modèle client-serveur doit maintenir et exécuter toutes les tâches de gestion. Un lien sur le réseau est ouvert pour chaque tâche de gestion pour la durée de la tâche. Pour le modèle avec agents mobiles, les tâches sont dirigées sur les composantes à gérer permettant ainsi une moins grande utilisation des liens et un parallélisme d'exécution. Les agents mobiles seront donc de toute évidence au centre de notre architecture.

### 3.2.3 Technologies pour l'accès aux ressources

Pour mener à bien une tâche de gestion, il est proposé d'utiliser les protocoles et outils de gestion disponibles sur les composantes. Nous ne préconisons pas l'approche utilisant uniquement des interfaces uniformes sur toutes les composantes. Cette approche ne permet pas la flexibilité recherchée pour effectuer des tâches diversifiées, complexes et spécifiques. Il est nécessaire d'avoir une architecture qui peut être adaptée à plusieurs environnements de gestion et pouvant intégrer rapidement les nouvelles technologies et composantes. De plus, il n'est pas encore possible d'avoir des réseaux ayant uniquement des composantes pouvant exécuter des agents mobiles. De tels réseaux, dont les réseaux actifs font partie, ne sont pas encore disponibles et prêts à remplacer les réseaux actuels. Nous voyons néanmoins l'avantage de créer un accès uniforme à un grand nombre de fonctions de base pour optimiser la taille de l'agent mobile de gestion et faciliter la création de nouveaux agents mobiles de gestion. Cependant, ces derniers peuvent demeurer conscients du type de composante qu'ils gèrent et cela leur donne plus de flexibilité pour effectuer des tâches plus spécifiques. Les agents mobiles détiennent l'intelligence nécessaire pour utiliser une grande gamme d'interfaces de programmation et de protocoles de gestion locaux pour mener à bien leurs tâches de gestion. Ils utilisent les outils de gestion mis à leur disposition sur chaque station de gestion. Cette intelligence augmentera nécessairement la taille des agents mobiles. Cependant, cela se fait au profit de la simplicité et de la possibilité d'inclure rapidement les agents mobiles dans la gestion de réseaux réels. Pour éviter que l'agent mobile de gestion ait une trop grande taille, l'architecture prévoit le chargement dynamique des fonctions de base en utilisant au choix, soit un proxy vers des agents stationnaires locaux ou encore en pré-installant le code sur chaque station de gestion pertinente. Ces principes sont explicités davantage dans la section 3.3.1. Nous savons que les communications tendent constamment à devenir plus rapides. Les contraintes de taille des agents mobiles peuvent donc être relâchées légèrement, en autant que les déplacements de ces agents soient bien orchestrés. Nous croyons que de créer plusieurs agents simples et spécialisés n'est pas une solution en soit pour réduire

le transfert de code. Le transfert continuel de connaissances et les fréquents échanges dus au manque d'autonomie d'un agent risquent en effet d'engendrer des communications inutiles.

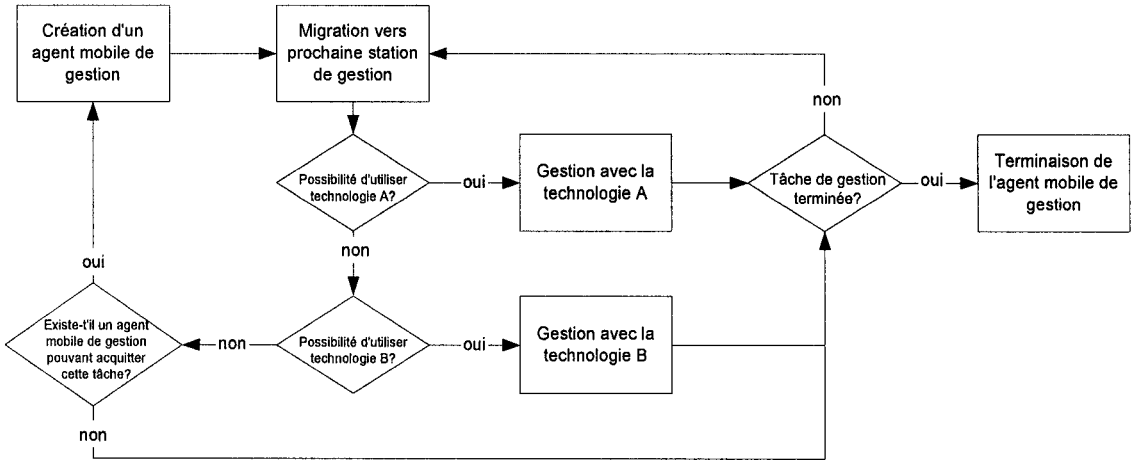


**Figure 3.4 Exemple de la connaissance d'un agent mobile de gestion**

Pour illustrer la connaissance d'un agent mobile de notre architecture, il convient de regarder la Figure 3.4 démontrant un agent mobile effectuant une tâche de gestion quelconque. L'agent mobile de cet exemple a la possibilité d'utiliser deux technologies différentes. Dans le cas où l'utilisation de ces deux technologies échouerait, il lui est possible de faire appel à un autre agent mobile pouvant effectuer la même tâche, mais dans un contexte technologique différent. Cette possibilité permet une plus grande flexibilité. Il est cependant primordial qu'un agent mobile dispose d'un large éventail de capacités pour minimiser ce genre d'interaction. Pour diminuer sa taille, l'agent mobile peut éviter de transporter le code des classes permettant l'utilisation de ces technologies. Au besoin, ces classes peuvent être installées localement de façon permanente par un mécanisme de chargement dynamique sur demande. Il est aussi possible, si aucun agent mobile disponible ne peut s'acquitter de la tâche, de simplement signaler cette impossibilité pour obtenir l'attention d'un administrateur sur ce problème. Ce manque pourrait commander une mise à jour de l'agent pour tenir compte de la déficience.

Dans cet ouvrage, un agent mobile de gestion utilisant des mécanismes globaux de gestion sera nommé agent mobile général. Si l'agent utilise des moyens de gestion plus spécifiques, il sera présenté comme spécialisé.

### 3.2.4 Multi-agents

Éclairé par la revue de littérature, il demeure toujours difficile de déterminer le nombre d'agents mobiles optimal dans un réseau. Cette tâche est compliquée par le fait que la détermination d'un niveau optimal pour une application ne règle pas le problème pour d'autres applications. De plus, la notion même d'optimalité est difficile à déterminer. Elle diffère selon l'effet que l'on recherche. Dans notre cas, nous voulons une performance acceptable mais aussi la simplicité d'utilisation. Notre modèle ne peut se satisfaire d'un seul agent mobile effectuant toutes les tâches de gestion. Ce cas serait totalement inefficace. Néanmoins, nous croyons qu'un bon compromis est de créer typiquement un ou deux agents mobiles pour chaque tâche de gestion. Au besoin, plusieurs agents mobiles différents peuvent être assignés à une tâche exigeante, si cela en améliore les performances. La construction de tels agents sera facilitée par la caractéristique modulaire de l'architecture que nous verrons à la section 3.3.1.

### 3.2.5 Intelligence de l'agent mobile de gestion

L'agent mobile de l'architecture sera dédié à des tâches de gestion qui prennent un laps de temps beaucoup plus long étant donné qu'elles sont effectuées par des humains et que parfois des limites physiques et départementales séparent les différentes parties du réseau. L'architecture inclut toute la gamme des tâches de gestion qui peuvent être décrites par une procédure précise. La Figure 3.5 montre un schéma de décision simple qu'un agent mobile peut suivre pour déceler un problème de communication. Chaque diagnostic de cette figure peut être fait sur des composantes hétérogènes à l'aide de plusieurs technologies.

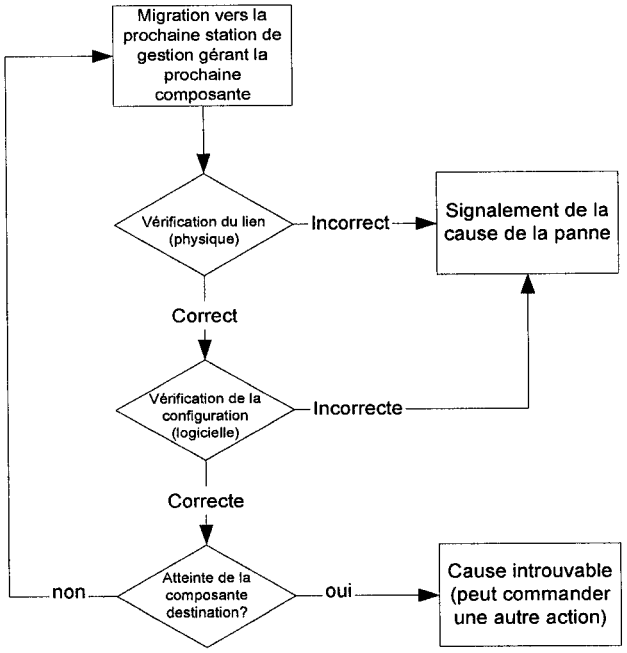


**Figure 3.5 Exemple d'un diagnostic simpliste procédural**

Il n'est donc pas question de système à apprentissage, mais d'un système expert de gestion à base d'agents mobiles. Ce système expert peut être programmé au fur et à mesure que les besoins surviennent. Chaque fois qu'un opérateur doit prendre un certain temps pour effectuer une tâche récurrente, il pourrait être intéressant d'utiliser ou de construire un agent mobile dédié à ce problème ou à cette classe de problèmes pour effectuer une tâche. Le nombre de tâches que l'ensemble des agents mobiles pourra effectuer sera déterminé à un instant donné par leur programmation. Ce modèle expert a l'avantage d'être plus simple à implémenter, considérant que l'on dispose de l'expertise, que le modèle à apprentissage. Il permet aussi de prévoir plus facilement le comportement de l'agent mobile. Le but n'est pas d'éliminer la présence humaine dans un réseau ou de gérer le réseau entièrement avec des agents mobiles, mais bien de réduire les tâches récurrentes, longues et procédurales. Avec le perfectionnement graduel du système expert, et comme notre modèle a comme requis d'être modulaire et facile à modifier, le système s'améliorera avec le temps. L'intelligence de l'agent

mobile de gestion pourrait fort bien reposer sur un autre principe qu'un système expert. Cependant, dans ce mémoire, ce ne sera pas le cas.

### 3.2.6 Topologie partielle et proximité

Nous avons jusqu'à maintenant considéré l'agent mobile et sa capacité à gérer plusieurs composantes. Il est maintenant essentiel de considérer la connaissance d'un tel agent de son environnement, qui est limité en ressources, et de son moyen d'action sur celui-ci. La Figure 3.6 illustre de manière générale l'architecture de gestion de la quasi-majorité des réseaux hétérogènes actuels à grande échelle. La gestion d'un réseau, même de dimension nationale, se fait majoritairement à partir d'une même station de gestion distante.

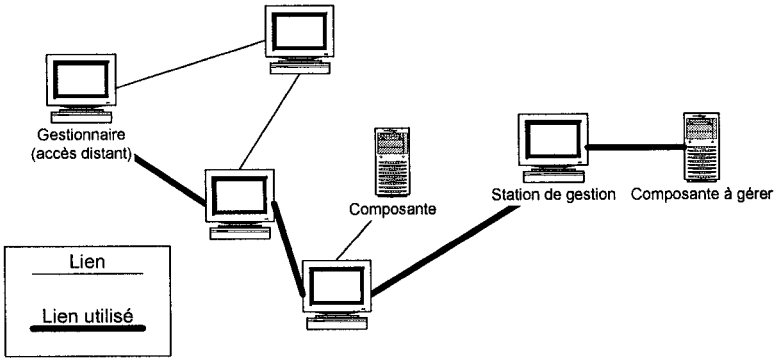


**Figure 3.6 Illustration de la gestion d'une composante à distance**

Étant donné l'impossibilité d'exécuter des agents mobiles sur toutes les composantes, et désirant éviter une gestion à distance inefficace, il devient nécessaire de tirer profit de la capacité de gestion à distance d'une composante tout en rapprochant la tâche de gestion le plus possible de cette composante. Ce principe est illustré à la Figure 3.7. À cette figure, le programme responsable de la gestion s'est déplacé le plus près possible de la composante à gérer, évitant ainsi une utilisation continuelle des liens de communication entre le gestionnaire et la composante à gérer. Les deux scénarios effectuent le même travail différemment. À la Figure 3.6, le gestionnaire exécute la tâche de gestion localement et effectue une gestion distante.

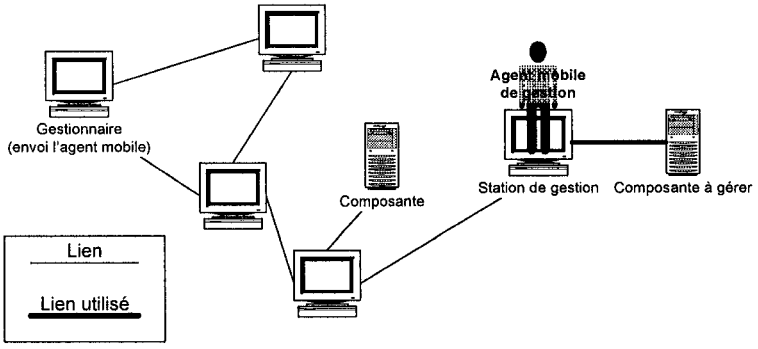


**Figure 3.7 Illustration de la gestion d'une composante à proximité**

À la Figure 3.7, celui-ci lance un agent mobile de gestion pour effectuer la tâche près de la composante. Le modèle utilisé pour l'architecture est celui de la gestion d'une composante à proximité.

Comme ce ne sont pas toutes les composantes qui peuvent accepter un agent mobile, il faut aussi connaître les emplacements des stations de gestion les plus adaptées pour une composante donnée. Le Tableau 3.1 donne un exemple illustrant comment une station de gestion peut être liée à une composante à l'intérieur d'une table. Cette table peut être comparée à un serveur de noms simples à la différence que, lorsqu'un agent mobile de gestion envoie le nom d'une composante, il reçoit le nom de la station de gestion responsable. Notez que si une composante a la capacité d'accueillir des agents mobiles de gestion, alors cette table retournera le nom de cette composante (par exemple, la composante D).

**Tableau 3.1 Exemple d'une table d'associations entre composantes et stations de gestion**

| Nom | Station de gestion |
|---|---|
| Composante A | Station de gestion 1 |
| Composante B | Station de gestion 2 |
| Composante C | Station de gestion 2 |
| *Composante D* | *Composante D* |
| Composante E | Station de gestion 3 |

L'agent mobile n'a qu'à se référer à cette table pour connaître où il pourra poursuivre sa tâche de gestion s'il veut gérer une composante. Cette table peut être soit dans sa

mémoire soit installée sur un ou plusieurs serveurs que l'agent peut interroger au besoin. Elle mémorise diverses informations sur les composantes comme leur type, leur adresse, leur protocole et leur nom. Présentement, elle est statique, ce qui indique que nous utiliserons toujours une même station de gestion pour une même composante, et ce, avec comme contrainte que la station doit être le plus près possible de la composante. La proximité se définira par le nombre de sauts entre la composante et la station de gestion. Si la station de gestion n'est pas disponible, la gestion se fera sur la station de gestion courante ou sur une autre station de gestion, selon la connaissance de l'agent mobile de gestion.

Maintenant que l'agent mobile peut lier une station de gestion à une composante, il faut aussi que l'agent ait une idée de la topologie du réseau qu'il a à gérer. Le cas extrême serait de programmer l'agent avec la topologie complète du réseau à gérer. De toute évidence, cette programmation est très peu évolutive et demande beaucoup de mémoire. Il faut donner un moyen à l'agent mobile de gestion d'obtenir cette topologie. L'architecture proposée ne requiert pas cette connaissance totale de la topologie et ne demande que d'avoir une vision partielle de la topologie. Cette vision partielle peut être requise étant donné l'hétérogénéité du réseau qui pourrait rendre impossible la découverte complète du réseau par interrogation des composantes une à une. Cette connaissance n'est cependant pas requise pour tous les types de réseaux. Par exemple, un agent mobile de gestion qui n'effectue qu'une gestion générale dans un réseau IP ne requiert pas cette connaissance. La Figure 3.8 montre un exemple de ce principe en utilisant plusieurs nuages.

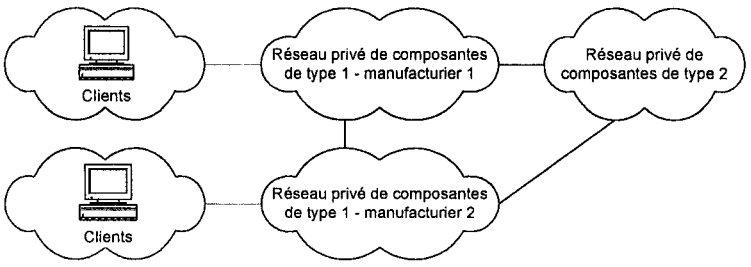


**Figure 3.8 Exemple de la connaissance de la topologie d'un agent mobile**

Les nuages représentent des sous-réseaux logiques composés de plusieurs composantes à technologies semblables. L'agent mobile n'a qu'à connaître la topologie partielle du réseau qu'il a à gérer. L'ajout d'une composante dans un nuage est immédiatement pris en compte par l'agent mobile, car ce dernier découvrira la topologie d'un nuage par l'interrogation des interfaces interconnectant ces topologies. À chaque identification, ce même agent peut consulter la table de liens (ou table de routage) entre composantes et stations de gestion pour être en mesure de migrer vers la prochaine station de gestion. Certains réseaux et certaines tâches peuvent très bien fonctionner sans donner à l'agent mobile une quelconque topologie. Pour les autres cas, l'agent mobile se doit de connaître la topologie partielle, c'est-à-dire les liens entre les nuages, pour savoir comment se diriger dans le réseau et pour savoir où accomplir une tâche de gestion. De plus, de tels agents pourraient avoir besoin de connaître les technologies qu'ils rencontreront à l'intérieur des nuages pour pouvoir interagir de façon appropriée avec les composantes.

### 3.2.7 Composition d'un agent mobile de l'architecture

Maintenant que l'architecture a été présentée, il convient de résumer la représentation abstraite d'un agent mobile du système de gestion (Figure 3.9).

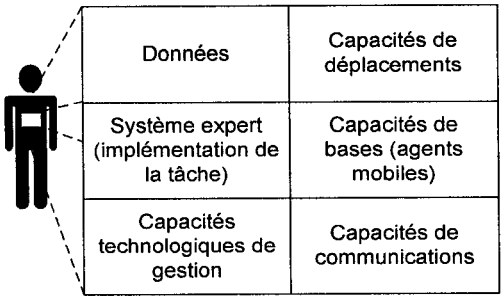


**Figure 3.9 Constitution d'un agent mobile de l'architecture proposée**

Cet agent mobile dispose des trois aspects ordinairement attribués aux agents mobiles : le déplacement, l'intelligence et les données. L'agent mobile de gestion ne fait pas exception. Plus spécifiquement, cet agent dispose de capacités de communication et de

capacités de gestion de réseaux basées sur plusieurs technologies. Il lui est aussi possible de se déplacer de station de gestion en station de gestion et d'hériter des fonctionnalités de base de tous les agents mobiles. Il transporte un certain nombre de données sur sa tâche, son état et ses objectifs. L'intelligence de cet agent est caractérisée par un système expert déterministe qui implémente l'algorithme de décision pour une tâche de gestion donnée.

### 3.2.8 Activation des agents mobiles de gestion et régulation de la population

L'activation des agents mobiles de gestion peut être effectuée par trois moyens : la personne constatant le besoin, un gestionnaire local ou un agent mobile chargé de surveiller et de vérifier le réseau. Dans un réseau de grande taille, le nombre d'agents mobiles lancé peut rapidement croître à un seuil inacceptable. L'architecture proposée limite déjà ce nombre en créant des agents intelligents autosuffisants. Le système de régulation de la population peut limiter l'envoi d'agents mobiles identiques. Cette protection est inspirée de la forte probabilité qu'un problème ou un besoin soit perçu par plusieurs composantes.

Des agents mobiles de gestion, dont la spécialité est la surveillance, peuvent parcourir le réseau et vérifier certains paramètres d'intérêt et activer les agents mobiles de gestion appropriés au besoin. Cette surveillance est laissée à la discrétion de l'administrateur du réseau et l'architecture fournie les outils pour créer de tels agents.

## 3.3 Caractérisation du réseau à gérer et du réseau de gestion

Cette section donnera des détails plus précis sur l'architecture proposée. Elle permettra de clarifier les différents concepts de l'architecture.

### 3.3.1 Modularité, habiletés et communications

L'architecture n'utilise pas qu'un seul agent mobile. Elle admet donc la possibilité de communications entre plusieurs agents. Il a déjà été mentionné que

l'architecture doit être simple. Cette simplicité est possible si les outils et les capacités servant à programmer un agent mobile sont bien définis en modules séparés et simples. De plus, l'utilisation d'un jeu de fonctions de base uniforme facilite la construction des agents mobiles de gestion. Le fait d'utiliser typiquement un ou deux agents mobiles par tâche diminue de beaucoup les communications. Cependant, il ne permet pas de les éliminer complètement et ces dernières demeurent essentielles à l'architecture.

Une approche modulaire, comme à la Figure 3.10, permet de créer diverses spécialisations, capacités et fonctionnalités qui peuvent être incluses dans différents agents mobiles.

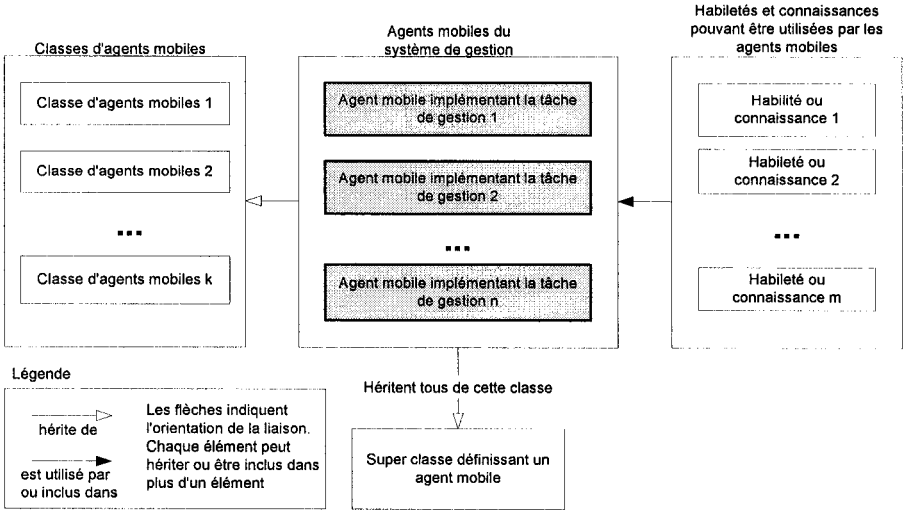


**Figure 3.10 Modèle modulaire des agents mobiles du système de gestion**

Il est alors simple d'éviter d'inclure des modules superflus dans un agent mobile qui n'en a pas besoin. En élaborant plusieurs blocs, il est facile de construire rapidement des agents mobiles effectuant des tâches diversifiées.

À la Figure 3.10, les agents mobiles sont représentés par les rectangles gris pâles. Le système de gestion contient donc n agents mobiles de gestion implémentant autant de tâches de gestion spécifiques concrètes. Ces agents mobiles peuvent hériter de k classes abstraites représentant ni plus ni moins un classement des agents en divers domaines de gestion. Ce classement peut être en fonction d'un type de tâches de

gestion, d'une catégorie de la classification ISO ou toute autre séparation logique. Ces classes abstraites ont comme but d'éviter de recréer le travail de structuration pour une gamme d'agents mobiles proches mais différents. Il est donc possible de créer une interface ou des fonctionnalités communes à plusieurs agents mobiles. Par exemple, à la Figure 3.11, deux agents mobiles de surveillance héritent d'une classe « surveillance » qui pourrait, entre autres, implémenter des fonctions temporisatrices permettant l'éveil et le sommeil de l'agent entre deux mesures de performance.

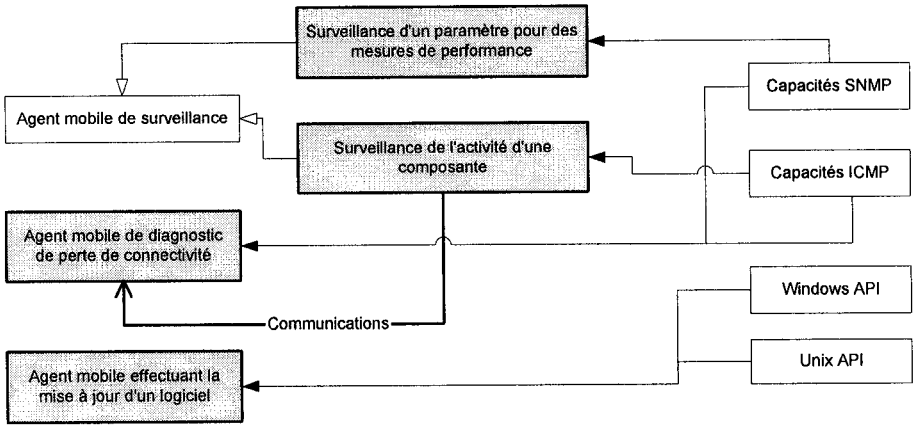


**Figure 3.11 Exemple d'un modèle modulaire**

Bien qu'ils représentent normalement aussi des classes, il est bon de placer les habiletés et connaissances des agents mobiles dans une catégorie à part. Ces modules correspondent aux interfaces avec les différents protocoles et systèmes de gestion du réseau. Les classes implémentant les habiletés et connaissances sont très importantes et ne doivent pas dépendre d'un agent mobile de gestion pour faciliter leur réutilisation et leurs mises à jour. Un agent mobile utilise plusieurs habiletés et peut hériter de plusieurs classes différentes. Bien entendu, chaque agent mobile hérite d'une super classe définissant un agent mobile, qui est fournie par la plate-forme d'agents mobiles choisie. L'architecture prévoit deux types d'habiletés : les habiletés pour gérer une composante et les habiletés utilisables à partir de la station de gestion. Par exemple, les premières représentent ce que l'agent peut faire avec une composante, alors que les secondes représentent ce que l'agent peut faire à partir de la station de gestion.

La Figure 3.11 illustre un exemple montrant notre approche modulaire et l'idée qu'un agent mobile peut compléter une tâche de gestion avec un minimum d'interactions avec d'autres agents mobiles. Il est donc possible de créer ou d'utiliser des modules permettant l'accès à certaines fonctionnalités telles que des protocoles de gestion ou des interfaces de programmation. Ensuite, lorsqu'un agent mobile est créé pour une nouvelle tâche de gestion, il peut utiliser les modules utiles à sa tâche. Il est aussi possible, comme pour l'agent de surveillance dans cet exemple, de créer des agents mobiles adaptés à une catégorie de tâches de gestion et de les spécialiser selon les besoins des différents types de surveillance.

À la Figure 3.11, nous avons inclus un exemple de communication où un agent de surveillance de l'activité d'une composante pourrait appeler l'agent mobile chargé du diagnostic. Ainsi, le premier agent n'a pas à transporter « l'intelligence » requise pour faire un diagnostic, mais peut quand même mener à bien la tâche de diagnostic en faisant appel à l'agent mobile qui a cette capacité.

La communication inter-agents n'est pas un élément de premier plan de l'architecture étant donné qu'en général une tâche de gestion est implémentée par un ou deux agents mobiles. Cependant, les communications conservent une place importante. Par exemple, un agent mobile de diagnostic pourrait se cloner et maintenir une communication entre clones pour vérifier conjointement plusieurs paramètres. Il pourrait être intéressant, advenant un réseau très hétérogène, d'envoyer des agents mobiles de diagnostic pour différents manufacturiers qui communiqueraient entre eux. Un agent mobile chargé de mesurer des performances pourrait aussi communiquer ses données statistiques à un agent mobile collecteur d'informations ou à une instance centrale. Pour simplifier le modèle et éviter l'apparition d'un système d'agents mobiles utilisant constamment le réseau pour leurs communications, nous limitons le nombre des interactions, sans toutefois les bannir. L'architecture se veut volontairement flexible sur ce point pour permettre toute sorte d'amélioration faisant appel aux communications.

Pour diminuer la taille des agents mobiles, l'architecture utilise deux principes différents. Le premier principe permet à l'agent mobile de charger dynamiquement des habiletés (modules) au besoin. Il suffit donc d'encapsuler une habileté à l'intérieur d'un agent stationnaire capable de répondre à des appels de procédures distantes. Un tel agent peut être téléchargé sur une station à la première utilisation et ensuite persister localement pour des utilisations futures. L'agent mobile peut donc référer aux classes implémentant ces habiletés comme si elles étaient incluses dans l'agent via un proxy. De cette manière, les agents mobiles conservent une faible taille, les classes utilitaires ne sont téléchargées qu'une seule fois au besoin pour chaque station de gestion et l'utilisation des ressources est plus optimale. L'utilisation de ces classes peut être totalement transparente au programmeur. Ce principe est illustré à la Figure 3.12.

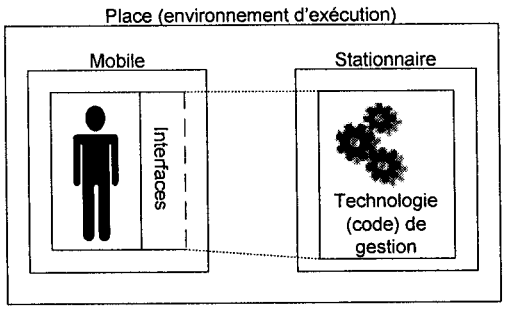


**Figure 3.12 Utilisation du code technologique par proxy**

L'agent mobile de gestion représenté par le personnage utilise le code de gestion par l'intermédiaire d'une interface de gestion, limitant ainsi le code transporté par l'agent. Le second principe, s'il n'est pas nécessaire d'avoir une architecture dynamique, consiste simplement à installer le code des agents mobiles de gestion sur les différentes stations de gestion pour éviter le transport de ce code. Les agents mobiles sont ainsi limités à la mobilité de leurs états et données. Le temps de réponse est amélioré étant donné une plus faible utilisation des proxy et une diminution du trafic, mais il devient plus difficile de modifier l'architecture pendant qu'elle est en marche. De plus, cette dernière méthode exclut la possibilité de conserver des informations, de façon similaire à une mémoire cache, sur une station de gestion. Les composantes de base de

l'architecture sont installées sur chaque station de gestion. Il est possible d'utiliser une approche hybride utilisant les deux méthodes d'accès.

En plus des communications par proxy, l'architecture prévoit des mécanismes de communications inter-agents standardisés et bien définis. Ce choix est motivé par l'étendue des possibilités de celle-ci qui pourrait laisser la responsabilité de créer certains modules technologiques à des manufacturiers ou à des tierces parties. Il est aussi motivé par l'ouverture de l'architecture et le requis de flexibilité et de compatibilité. De plus, les agents mobiles de gestion pourraient être appelés à utiliser et divulguer des informations, prendre part à des négociations, exécuter des actions sur demande et gérer les erreurs pouvant survenir non pas seulement sur le réseau, mais dans le fonctionnement même des agents mobiles. L'architecture propose l'utilisation de mécanismes inspirés de standards tels que FIPA et de langages tels que KQML (Knowledge Query and Manipulation Language).

### 3.3.2 Réseau à gérer

Les agents mobiles de gestion de l'architecture se déplacent dans un réseau restreint et privé qui peut inclure les utilisateurs de ce réseau. Les agents ne dépassent jamais ce cadre. La Figure 3.13 montre un réseau général incluant quelques familles de composantes réseau et les clients s'y attachant.

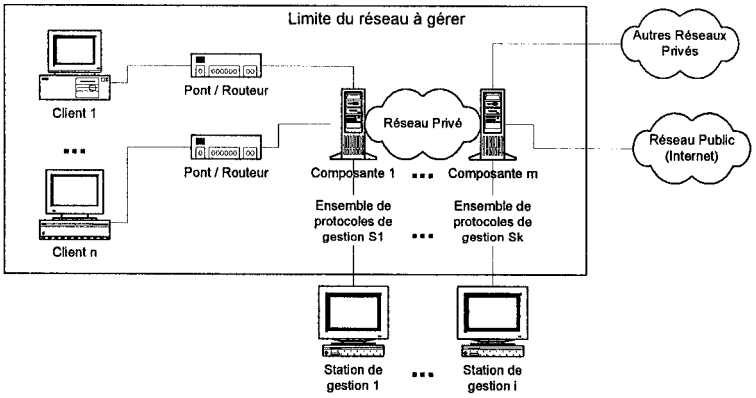


**Figure 3.13 Réseau géré par les agents mobiles**

Les éléments nommés « composantes » ne peuvent pas toujours exécuter une plate-forme d'agents mobiles ou tout autre logiciel ou agent complexe. Dans ce cas, elles sont donc gérées par des stations de gestion via un ensemble de protocoles ou d'interfaces de gestion. Sur cette illustration, les stations de gestion ne sont pas gérées par les agents mobiles et sont donc à part du réseau à gérer. Les clients sont habituellement des éléments moins sollicités par le réseau et peuvent généralement exécuter la plate-forme d'agents mobiles sans problèmes. Les agents mobiles ont pour tâches de gérer l'ensemble du réseau qui se trouve à l'intérieur du rectangle. Le nuage « Réseau Privé » modélise un réseau privé qui peut être constitué de plusieurs composantes hétérogènes et doit aussi être pris en compte par les agents mobiles de gestion. Ce réseau peut être relié à d'autres réseaux privés ou encore à des réseaux publics tel Internet. Ces derniers ne font pas partie du cadre de gestion et l'architecture ne permet en aucun cas ni l'arrivée ni l'envoi d'agents mobiles de gestion vers ces réseaux. La raison est évidente : la sécurité. Il faut noter que les clients peuvent aussi être liés par des réseaux locaux que l'agent mobile de gestion pourrait avoir à gérer.

La Figure 3.13 constitue une représentation du réseau géré par l'architecture de gestion. Cependant, il ne démontre aucunement le chemin emprunté par les agents mobiles. Ce chemin peut être différent à cause d'un certain nombre de contraintes physiques et logiques. Nous verrons ce détail dans la section 3.3.3.

### 3.3.3 Environnement d'exécution des agents mobiles de gestion

L'architecture tient compte de l'impossibilité actuelle de déplacer les agents mobiles sur tous les équipements du réseau. Rien ne semble indiquer, à court terme, que toutes les composantes pourront exécuter un environnement Java sur lequel la quasi-totalité des plates-formes d'agents mobiles reposent. Il faut cependant noter que de telles composantes existent déjà et que l'architecture proposée tire profit des équipements pouvant exécuter sans problèmes ces plates-formes. En effet, ces derniers reçoivent directement les agents mobiles. Nous considérons qu'une composante peut exécuter une plate-forme d'agents mobiles sans problèmes si cela ne nuit pas

considérablement à ses performances, et, de toute évidence, si la composante peut installer cette plate-forme. Les composantes ne permettant pas l'exécution d'agents mobiles sont gérées par une station de gestion dédiée selon le principe de proximité, introduit dans la section 3.2.6. Les agents mobiles ont donc un impact minime sur les composantes cruciales du réseau à gérer. Ces stations de gestion sont donc obligatoires pour que le modèle fonctionne avec tous les types de composantes disponibles. Elles font généralement déjà partie des réseaux réels, permettant d'adopter notre modèle sans devoir investir largement dans de nouveaux équipements coûteux. Les agents mobiles requièrent donc un réseau de gestion différent de celui du réseau géré qui est illustré à la Figure 3.14.

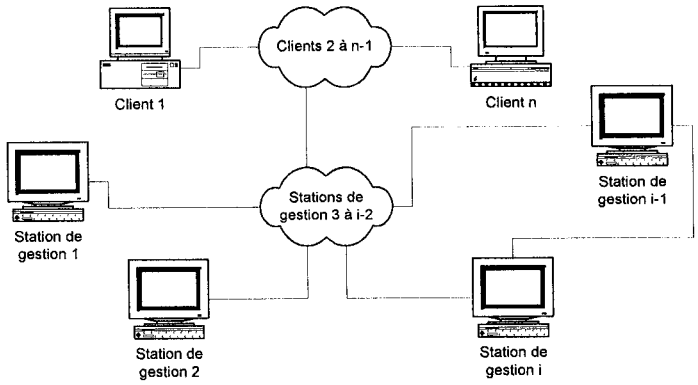


**Figure 3.14 Réseau parallèle (réseau de gestion) pour les agents mobiles**

En d'autres mots, les agents mobiles se déplacent sur les stations de gestion dédiées aux composantes ayant une capacité de programmation restreinte et sur les composantes capables d'exécuter les plates-formes d'agents mobiles. Contrairement à la Figure 3.13, le réseau où les agents mobiles se déplacent (Figure 3.14) n'inclut pas toutes les composantes. La Figure 3.14 ne représente que le réseau physique des agents mobiles sur lequel ils pourront s'exécuter et considère le cas où seuls les clients et les stations de gestion peuvent exécuter des agents mobiles. Pour un réseau où certaines composantes peuvent accepter des agents mobiles, nous verrions ces dernières apparaître dans le réseau parallèle.

Pour bien comprendre comment ces deux réseaux interagissent ensemble, il convient d'introduire un modèle à deux couches qui est illustré à la Figure 3.15.

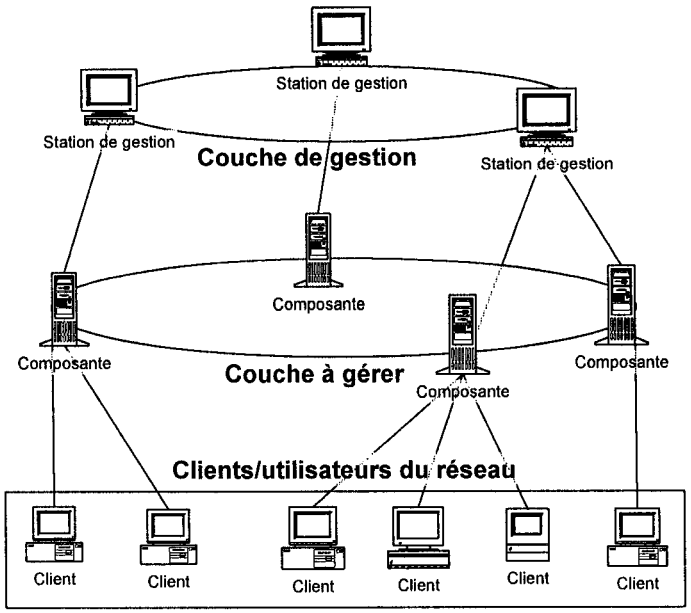


**Figure 3.15 Architecture générale du réseau privé à gérer et de gestion**

Ce modèle inclut une partie des éléments de la Figure 3.13 et de la Figure 3.14. Il permet de voir clairement la distinction entre composante gérée et station de gestion. Ces dernières font partie de la couche de gestion et ont comme seul rôle la gestion des premières. La couche à gérer comprend tous les éléments qui seront gérés par notre système de gestion. Cela inclut toutes les composantes du réseau à l'intérieur du rectangle de la Figure 3.13. Les clients font partie de ces deux couches et devraient toucher aux deux couches sur le dessin. Cela n'a pas été dessiné par souci de clarté. De plus, il faut ajouter qu'une composante pouvant exécuter une plate-forme d'agents mobiles pourrait aussi être membre de ces deux couches.

### 3.3.4 Plan d'architecture

Nous avons déjà mentionné la modularité, les habiletés et le modèle de communication de l'architecture. Il a aussi été question du réseau à gérer et du réseau

de gestion. Nous avons présenté ces éléments sur une échelle globale. Il convient donc maintenant d'introduire un plan général ainsi qu'un plan de chacun de ces aspects de l'architecture proposée pour bien en comprendre la constitution.

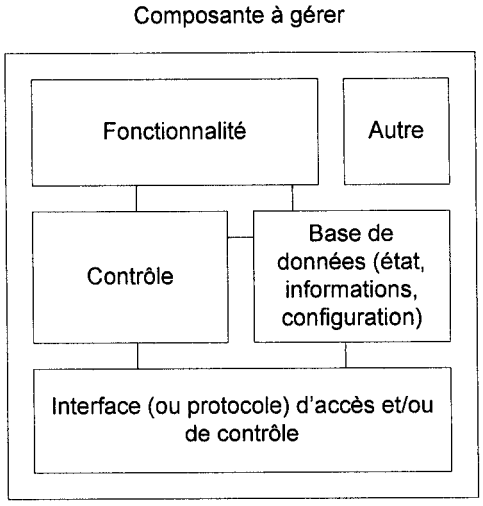


**Figure 3.16 Modèle général d'une composante à gérer**

À la Figure 3.16, nous avons la représentation d'une composante à gérer. Une telle composante est pourvue d'une ou de plusieurs interfaces permettant le contrôle et l'accès à une forme de base de données. La composante à gérer a sa fonctionnalité propre et peut aussi être constituée d'autres éléments dont, sans s'y limiter, un environnement d'exécution d'agents mobiles.

La Figure 3.17 introduit le modèle d'une station de gestion de l'architecture proposée. Les caractéristiques d'une station de gestion sont les suivantes : elle dispose généralement d'un système d'exploitation et de librairie plus ou moins spécifique à ce système d'exploitation. Elle dispose aussi d'un certain éventail de moyens d'accès aux ressources qu'elle doit gérer. Dans le cas de notre architecture, la station de gestion dispose obligatoirement d'une plate-forme d'agents mobiles ayant un environnement d'exécution et une banque d'agents mobiles de gestion. Il est à noter qu'une station de gestion pourrait se gérer elle-même, ce qui intégrerait le modèle de Figure 3.16 à celui de la Figure 3.17.

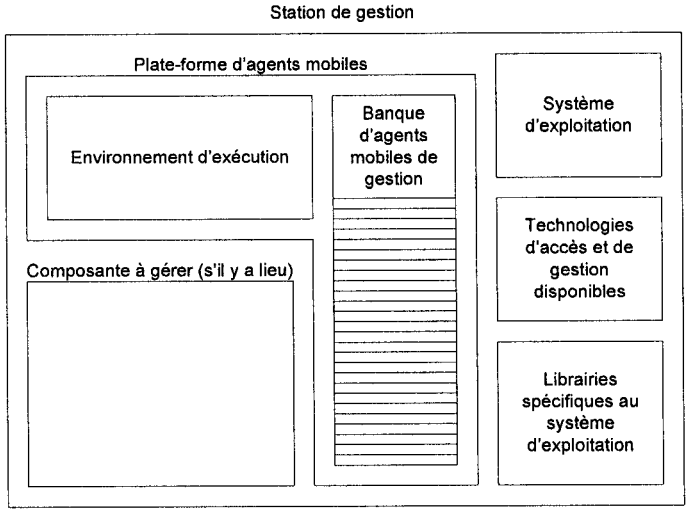


**Figure 3.17 Modèle général d'une station de gestion**

La plate-forme d'agents mobiles peut être détaillée davantage. À la Figure 3.18, on voit que la table d'associations entre composantes à gérer et stations de gestion est un agent mobile (ou stationnaire) qui peut rester de façon permanente sur chaque station de gestion. De plus, les librairies et fonctionnalités propres à la plate-forme sont introduites pour démontrer que les agents mobiles utilisent d'autres moyens d'action en plus des technologies d'accès et de gestion, et des librairies spécifiques au système d'exploitation.

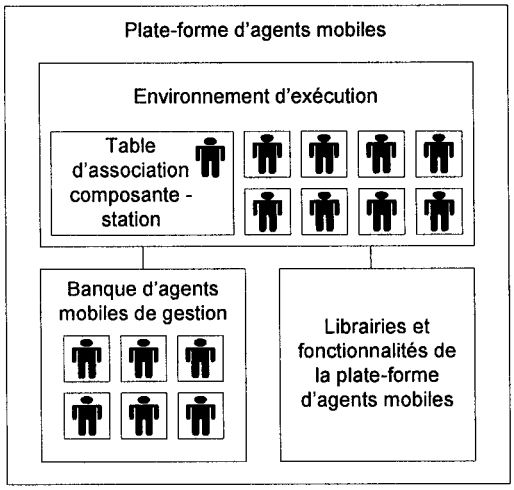


**Figure 3.18 Modèle de la plate-forme d'agents mobiles**

Pour illustrer tous les concepts et leurs relations introduits dans ce chapitre, il convient de regarder la Figure 3.19. Dans cette figure, on voit les liens de gestion et les liens utiles. Les deux cas possibles de gestion d'une composante sont représentés, soit la composante qui peut être une station de gestion et la composante qui doit être gérée à distance par une station de gestion. Des résumés des contenus des trois entités principales sont représentés par un agrandissement.

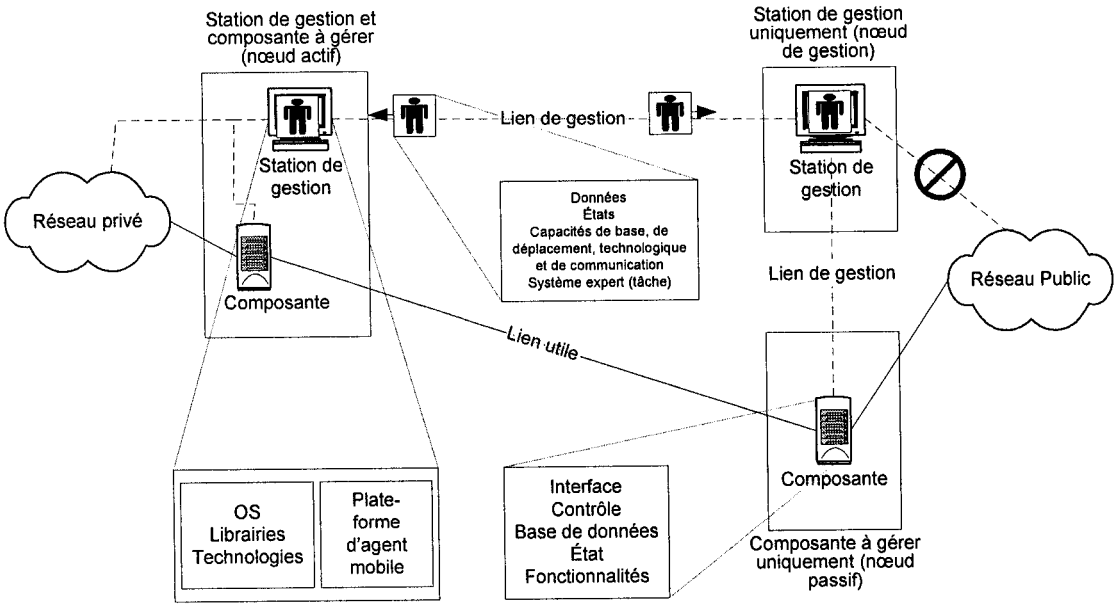


**Figure 3.19 Plan général démontrant les concepts de l'architecture et leurs liens**

Nous voyons aussi la représentation des réseaux privés et publics par des nuages, indiquant la connectivité et l'étendue de ce réseau dans un format général.

Un dernier aspect majeur n'a pas encore été explicité: la sécurité. À la Figure 3.19, nous voyons un premier mécanisme de sécurité par l'interdiction d'avoir un lien de gestion sortant vers un réseau public.

### 3.3.5 Aspects de sécurité

Le cadre restreint défini dans la section 3.3.2 permet de poser l'hypothèse que le réseau ne peut être menacé par des attaques extérieures au réseau. Les agents mobiles

ne sont lancés qu'à partir du réseau privé. L'architecture proposée repose sur l'hypothèse que le réseau privé, hormis les clients, peut être considéré sans hôtes malicieux et relève d'un contrôle suffisamment restreint pour avoir le niveau de confiance requis. En fait, les seules faiblesses de ce réseau privé pourraient être ses utilisateurs (clients). Ces derniers doivent donc être soit incapables de lancer un agent mobile sur le réseau privé ou soit limités à l'envoi d'agents mobiles bien définis. Autrement dit, un utilisateur normal ne peut créer son propre agent mobile de gestion. Par l'application de filtres entre ces utilisateurs et les composantes du réseau privé, il devient possible de s'assurer que ce qui circule sur le réseau est un agent approuvé par son administration. Pour ce qui est de la protection des informations confidentielles telles que des mots de passe, il faut absolument que tout agent mobile qui quitte les limites de ce réseau privé pour entrer sur une station dédiée à un utilisateur normal soit exempt de données confidentielles. Le filtre peut donc agir dans un sens comme dans l'autre. Aussi, pour contrôler le nombre d'agents mobiles lancés sur le réseau, ce qui peut être une attaque dans la classe des dénis de service, l'architecture ne permet aux utilisateurs normaux de lancer un agent mobile que sur autorisation. Bref, l'architecture proposée évite donc à la fois l'envoi d'agents mobiles modifiés et malicieux, mais aussi l'envoi déraisonnable d'agents mobiles. Au besoin, nous pourrions utiliser des quotas pour laisser plus de liberté à certains utilisateurs.

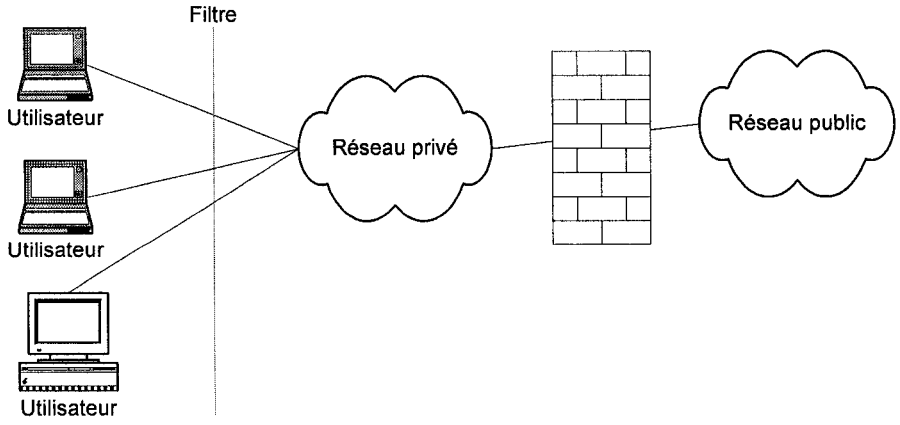


**Figure 3.20 Modèle de sécurité de l'architecture**

La Figure 3.20 illustre les différents concepts de sécurité de l'architecture. Il est à noter que le mur n'empêche que les agents mobiles extérieurs de s'exécuter sur le réseau privé et non pas les communications de tout ordre. La ligne pointillée représente le filtre qui agit comme barrière perméable, mais sélective, entre utilisateurs normaux et un réseau privé. L'architecture prend donc en compte les problèmes de sécurité courants, mais ne peut pas résister à un sabotage provenant d'un administrateur ayant les privilèges requis pour lancer, modifier et créer un agent mobile.

## 3.4 Déplacements et algorithmes de gestion généraux

Dans cette section, nous verrons d'abord les modèles de déplacement utilisés dans l'architecture proposée. Ensuite, nous donnerons quatre algorithmes généraux illustrant des tâches de gestion pour cette architecture.

### 3.4.1 Déplacement des agents mobiles

Comme l'architecture permet toute tâche de gestion, le modèle de déplacement des agents mobiles préconisé n'est ni celui utilisant un itinéraire ni celui effectuant une diffusion (broadcast). En fait, ce choix dépend directement de la tâche à accomplir. Par exemple, le diagnostic de fautes dans un réseau favorisera le modèle itinéraire pour enquêter de nœuds en nœuds en évitant des communications inutiles. Le modèle « diffusion » a le potentiel de trouver la panne plus rapidement, mais au prix d'une consommation plus élevée de ressources sur le réseau. Par l'addition de plusieurs diagnostics simultanés, la somme des ressources utilisées pourrait devenir énorme. Dans le cas d'une tâche de surveillance, le modèle « diffusion » pourrait s'avérer plus intéressant pour obtenir une meilleure résolution. L'architecture proposée ne fixe pas le mode de déplacement et de communication des agents mobiles au profit d'une plus grande flexibilité pour la gestion. Ce déplacement peut être déterministe ou probabiliste selon les besoins. Il peut utiliser les tables de routage ou une connaissance préprogrammée.

### 3.4.2 Algorithmes généraux de gestion de réseaux

L'architecture ne pourra traiter que d'un nombre fixe de tâches de gestion et de problèmes à un instant donné. Ces tâches suivront toujours les mêmes procédures. Par contre, le modèle pourra facilement évoluer avec l'aide des opérateurs.

Il est impossible de fournir un algorithme général qui peut englober toutes les tâches de gestion. Par contre, nous pouvons tout de même donner sommairement certains algorithmes de tâches de gestion généraux. Ces algorithmes ne sont pas spécifiques et demanderaient à être adaptés pour différents réseaux, différents contextes et différentes technologies. Nous verrons des algorithmes précis sous forme de schémas dans le prochain chapitre. Néanmoins, ils présentent bien diverses tâches de gestion qui peuvent être envisagées dans le cadre de l'architecture proposée. Pour éviter d'alourdir le texte, quand nous écrirons qu'un agent mobile visite une composante, il est sous-entendu qu'il peut avoir à visiter la station de gestion de cette composante pour les raisons déjà énoncées. Le gestionnaire est la station initiant la tâche de gestion.

La Figure 3.21 démontre un cas où un agent mobile est affecté à chaque station de gestion gérant les composantes devant être surveillées. On obtient donc une résolution maximale étant donné qu'un agent surveille à tout moment toutes les composantes d'intérêt. Dans un premier temps, le gestionnaire spécifie les valeurs limites auxquelles l'agent mobile doit réagir ainsi que le temps approximatif entre les lectures. Les agents mobiles se déplacent près des composantes à gérer et prennent des lectures périodiques sur chacune d'entre-elles. Dans cet algorithme, s'il y a dépassement, l'agent mobile se clone et envoie ce dernier faire un rapport au gestionnaire et peut ainsi poursuivre la surveillance. Il serait aussi possible d'appeler un agent mobile spécialisé dans le problème détecté. Selon les besoins, cet algorithme peut être modifié pour arrêter de signaler les excès en arrêtant la surveillance pour éviter une prolifération d'agents mobiles rapportant ces excès. Aussi, dans l'éventualité d'une surveillance avec des contraintes de temps souples, il pourrait être plus avantageux de lancer moins d'agents mobiles que cet algorithme le prévoit.

**Gestionnaire :**

Obtention des composantes devant être surveillées (ensemble de nœuds ou tous)

Création de N (pour N stations de gestion gérant M composantes) agents mobiles appropriés pour la surveillance du trafic

Obtenir les valeurs de trafic qui devront provoquer une action de l'agent mobile

Obtenir la résolution voulue pour les lectures (t secondes entre lectures)

De i = 1 à N

    Envoi de l'agent mobile i vers la station de gestion i

**Agent mobile i :**

Tant que la surveillance est requise

    Pour toutes les composantes j = 1 à C (C composantes sont gérées par la station courante) gérées par cette station de gestion

        Lire la valeur de trafic

        Si la valeur de trafic lue excède la valeur limite spécifiée

            Cloner l'agent mobile i (agent créé sera nommé i,j)

            Envoyer agent mobile i,j vers la station de gestion

    Dormir t seconde(s)

Terminer l'agent mobile i

**Agent mobile i,j :**

Rapporter l'excès de trafic sur la composante j gérée par la station de gestion i

Terminer l'agent mobile i,j

**Figure 3.21 Surveillance du trafic de toutes les composantes d'un réseau**

La Figure 3.22 n'a comme but que de démontrer sommairement le diagnostic de problèmes de liaison. Nous insistons sur le fait que cet algorithme est sommaire, car le diagnostic de fautes peut être compliqué par une multitude de faits qui ne figurent pas sur cet algorithme. Dans cet algorithme, un gestionnaire est d'abord avisé d'un problème empêchant l'établissement d'une liaison entre deux nœuds distants. Il envoie donc l'agent mobile spécialisé pour ce type de problème en lui indiquant la source signalant le défaut ainsi que la destination qui est injoignable. L'agent mobile, dans l'exemple, peut donc tenter de remonter le chemin en effectuant une série de

diagnostics sur chaque nœud pour déterminer les problèmes possibles pour la liaison donnée.

**Gestionnaire :**

Signalement d'un problème de liaison entre deux composantes

Création d'un agent mobile spécialisé dans le diagnostic de problèmes de liaison

Obtenir le nom (ou l'adresse) de la composante signalant le problème

Obtenir le nom (ou l'adresse) de la composante de destination inaccessible

**Agent mobile :**

Migration vers la station de gestion liée à la composante ayant décelé le problème

Tant que l'agent mobile n'atteint pas la destination souhaitée

Interrogation de la composante (interfaces, protocoles, API présent) sur diverses couches réseaux (physique, application,...) et selon la mémoire accumulée de l'agent mobile

Si problème trouvé

Vérifier si ce problème empêche la liaison avec la composante suivante

Si oui

Migrer vers la prochaine station de gestion liée à la prochaine composante sur la liaison souhaitée (via le réseau parallèle)

Interrogation de la composante

Établissement d'un diagnostic selon la mémoire sur lequel des noœds empêchent la liaison

Retourner au gestionnaire et rapporter l'erreur

Terminer l'agent mobile

Sinon

Conserver le problème en mémoire

Migrer vers la prochaine station de gestion liée à la prochaine composante sur la liaison souhaitée

**Figure 3.22 Diagnostic d'un problème de liaison simple**

La façon dont l'agent mobile s'y prend et les défauts qu'il doit envisager sont fortement dépendants de la technologie et ne peuvent donc pas figurer dans cet algorithme. Bien entendu, le seul fait de déceler un défaut ne donne pas automatiquement la cause du problème. L'agent mobile doit conserver ce défaut en mémoire et poursuivre son diagnostic sur le nœud suivant. Dans l'algorithme, les conditions d'arrêt sont l'arrivée

à la destination ou l'impossibilité de trouver un chemin entre deux nœuds. La conclusion à tirer du premier cas est que l'erreur n'a pas été détectée, a été corrigée ou était une fausse erreur. Plusieurs décisions peuvent être prises selon l'implémentation : lancement d'un agent mobile ayant une autre spécialisation ou envoi d'un technicien pour vérifier ou diagnostiquer un problème en dehors du champ d'expertise des agents mobiles. Il faut toujours se rappeler que l'agent mobile ne voyage pas toujours sur les composantes, mais bien sur un réseau parallèle. Ainsi, le simple fait de se déplacer d'une station de gestion à une autre ne signifie pas que le lien est correct entre les composantes gérées par ces stations. L'agent mobile doit reconnaître les cas où la liaison est impossible à effectuer et ainsi en arriver à la conclusion que ce lien est défectueux à cause d'un problème sur ce nœud ou le suivant. L'avantage du réseau parallèle, c'est que l'agent, connaissant ce fait, peut interroger le nœud suivant pour obtenir plus de détails et ainsi éviter de diagnostiquer un problème sur le dernier nœud visible si l'erreur provient du nœud suivant. L'agent mobile de gestion peut aussi se cloner pour tenter d'initier une connexion entre deux composantes pour vérifier que la communication est possible entre deux nœuds.

Les algorithmes suivants démontrent comment une même tâche de gestion peut être différente selon les requis du gestionnaire. Les deux algorithmes produisent le même résultat final, mais utilisent deux modèles de propagation différents : un seul agent avec itinéraire et multi-agents avec diffusion.

Dans les algorithmes de la Figure 3.23 et de la Figure 3.24, le but est de mettre à jour une configuration sur un nombre présélectionné de composantes. Cette sélection peut être déterminée au départ ou en chemin. Dans le premier cas, le gestionnaire soumet une liste de composantes à visiter ou encore impose la visite et la modification de chaque composante. Dans le second cas, il est indiqué que la mise à jour se fait selon le besoin et l'agent mobile visite chaque composante et vérifie si elle doit être mise à jour. Dans l'algorithme de la Figure 3.23, il faut initialiser l'itinéraire de l'agent mobile. Contrairement à l'algorithme de la Figure 3.22, l'agent mobile doit visiter un nombre de nœuds bien défini et il est possible de tracer un chemin.

**Station de gestion :**

Création de l'agent mobile de mise à jour de configuration

Obtention des composantes devant être mises à jour (ensemble de nœuds, tous ou selon besoin)

Obtention des paramètres à modifier et des valeurs ou fonctions de modification

Initialisation de l'itinéraire de l'agent (stations de gestion liées aux composantes)

**Agent mobile :**

Tant qu'il reste des nœuds à visiter

    Vérification de la configuration courante

    Si besoin de mise à jour

        Application des modifications sur les paramètres visés

    Migration vers la prochaine station de gestion

**Figure 3.23 Mise à jour d'une configuration (économie de ressources par unité de temps)**

**Station de gestion :**

Obtention des composantes devant être mises à jour (ensemble de nœuds, tous ou selon besoin)

Création de N (pour N stations de gestion gérant M composantes) agents mobiles appropriés pour la surveillance du trafic

Obtention des paramètres à modifier et des valeurs ou fonctions de modification

De i = 1 à N

    Envoi de l'agent mobile i vers la station de gestion i

**Agent mobile i :**

Lire la valeur de trafic

Pour toutes les composantes (nommées j) gérées par cette station de gestion

    Vérification de la configuration courante

    Si besoin de mise à jour

        Application des modifications sur les paramètres visés

Terminer l'agent mobile i

**Figure 3.24 Mise à jour d'une configuration (meilleur temps de réponse)**

Bien que cette initialisation semble simple, elle constitue un autre problème en soi étant donné qu'elle peut se faire selon plusieurs principes : sans ordre, chemin optimal et

aléatoirement. Après cette initialisation, l'agent mobile utilisant l'algorithme de la Figure 3.23 doit parcourir chaque station de gestion liée aux composantes à gérer et doit vérifier s'il peut faire la mise à jour. Cette vérification peut être liée au besoin de faire la mise à jour ou encore à la possibilité technique de la faire. Par exemple, s'il faut modifier une entrée de base de données et qu'elle ne figure par sur la composante, l'agent mobile pourrait décider de la créer ou de ne rien faire. Suite à la vérification, l'agent mobile fait le changement puis migre vers le nœud suivant de son itinéraire. L'algorithme de la Figure 3.24 présente le même algorithme de mise à jour. Cependant, un agent est lancé pour chaque station de gestion, ce qui implique que ce dernier effectue un nombre limité de mises à jour. Ce nombre correspond au nombre de composantes que la station de gestion doit gérer. L'agent mobile meurt immédiatement après sa tâche.

Chaque algorithme ne traduit pas le principe que l'agent mobile doit composer avec un réseau hétérogène. Ainsi, il n'y a pas de « si la composante est de telle technologie » dans les algorithmes. Ces considérations devront effectivement être incluses dans les versions réelles des algorithmes des agents mobiles de gestion de cette architecture, à moins qu'ils n'utilisent des interfaces de gestion uniformes. De plus, pour éviter une prolifération d'agents mobiles, les algorithmes basés sur le modèle « diffusion » (Figure 3.21 et Figure 3.24) utilisent un agent mobile par station de gestion et non par composantes. Il faut considérer que la meilleure solution pourrait être d'utiliser plusieurs agents mobiles avec de petits itinéraires. Nous laissons cette considération, ainsi que celles reliées aux itinéraires optimaux, à la phase d'expérimentation.

S'il est nécessaire d'implémenter une nouvelle tâche de gestion, il suffit d'écrire un autre agent mobile et de l'utiliser au sein du réseau. De plus, par la nature même des agents mobiles, il devient aisé de modifier les tâches de gestion existantes. Il suffit de modifier leur programmation, de rappeler les anciennes versions et de réinstaller les nouvelles. Les plates-formes d'agents mobiles fournissent déjà ces moyens. Avec l'aide des agents mobiles, et avec leur constante évolution, il devient possible de

construire un système de gestion automatisé efficace étape par étape. Ainsi, tout comme un logiciel s'adapte graduellement à de nouveaux API (Application Program Interface), de nouveaux besoins, de nouvelles technologies et de nouveaux systèmes d'exploitation, les agents mobiles de gestion s'adaptent à de nouvelles composantes, de nouvelles technologies et de nouveaux outils de gestion de bas niveau.

# CHAPITRE 4
# IMPLÉMENTATION ET RÉSULTATS

Au chapitre précédent, nous avons décrit l'architecture de gestion proposée. Dans ce chapitre, nous présenterons les éléments qui ont été retenus pour évaluer l'efficacité, la pertinence et les performances de cette architecture. Notre but est d'éprouver l'architecture proposée dans un contexte réel et les tests choisis reflètent bien cet objectif. Nous présenterons l'implémentation des mécanismes de déplacements, de communications entre agents mobiles et agents stationnaires implémentant le code de gestion, ainsi que deux agents mobiles permettant la localisation de pannes et l'identification des causes de ces pannes. L'architecture de gestion est construite par-dessus la plate-forme d'agents mobiles commerciale Grasshopper. Finalement, nous présenterons les tests et les résultats pour l'architecture proposée suivi d'une discussion.

## 4.1 Implémentation de l'architecture

Dans cette section, nous présenterons les différents éléments de l'architecture qui ne sont pas des agents mobiles. En premier lieu, il sera question des classes représentant les principaux éléments de l'architecture. Ensuite, nous présenterons les interfaces de gestion uniformes ainsi que les modules technologiques implémentant ces interfaces. Finalement, il sera expliqué comment fonctionnent les tables d'associations entre composantes et station de gestion.

### 4.1.1 Classes de base

Cette section présente les classes de base de l'architecture qui ne sont ni des agents mobiles, ni des agents stationnaires. Pour les besoins de la discussion, les schémas sont simplifiés.

Les principaux concepts de l'architecture sont implémentés à l'aide de classes. Les composantes à gérer sont représentées par une classe nommée « Composante » et les stations de gestion sont représentées par la classe « Station ». Cette dernière classe hérite des fonctionnalités de la classe « Composante », car une station de gestion peut aussi être gérée. La classe « Composante » détient les informations sur les adresses, les numéros des interfaces liées à ces adresses, le nom et les habiletés de gestion de la composante. Cette dernière information permet de guider l'agent mobile de gestion dans sa recherche d'un moyen de gestion approprié pour une composante donnée. La classe « Station » a comme particularité de pouvoir attacher une composante à gérer et retient l'adresse « Grasshopper » de la composante. Ces deux classes sont schématisées à la Figure 4.1.

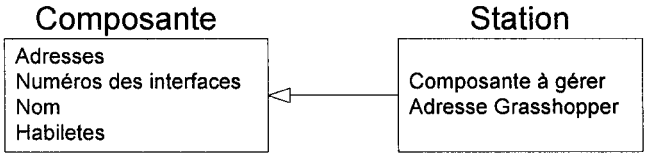


**Figure 4.1 Classes « Composante » et « Station »**

Une classe nommée « StructureRoutage » permet de structurer l'information requise pour décrire un point précis sur une composante. Cette structure conserve des informations sur l'état d'une interface matérielle précise. Elle permet aussi d'attacher de l'information supplémentaire à la discrétion de l'agent mobile de gestion.

Quatre classes permettent d'implémenter différents modèles de migration. Ces quatre classes héritent d'une même classe qui permet de conserver en mémoire les composantes visitées et à visiter. Les modèles de migration disponibles sont : séquentiel, indéterminé, aléatoire et diffusion (broadcast). Il est donc possible, sauf dans le cas indéterminé, de laisser le soin à ces classes de choisir les prochaines destinations. À la Figure 4.2, on constate les différents comportements de chacune de ces classes en regardant la description de la fonction « Suivant ». Le cas indéterminé retourne « null » étant donné que l'agent mobile ne suit pas un modèle de migration

déterminé. En effet, il utilise d'autres moyens pour déterminer les composantes suivantes qui ne sont pas supportés par la famille de classes « Déplacement ».

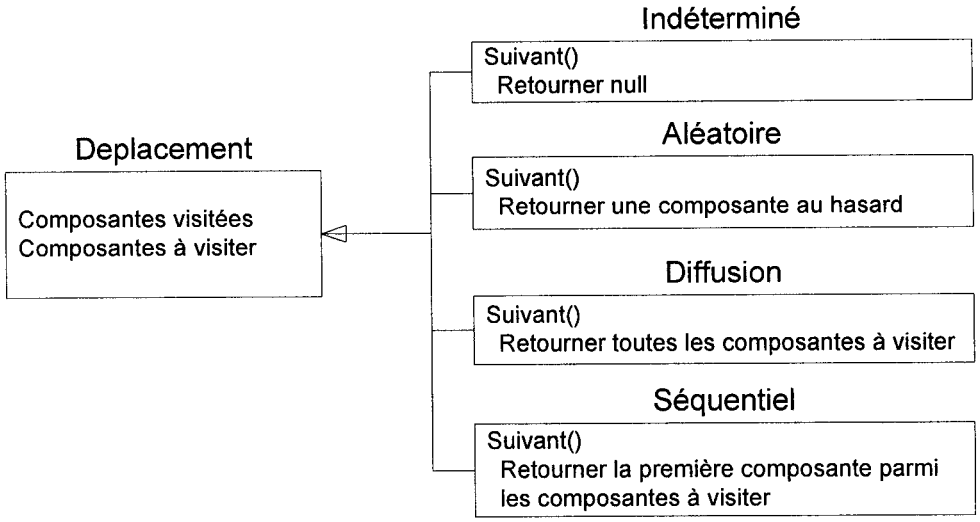


**Figure 4.2 Classes de déplacements**

Pour l'agent mobile chargé de diagnostic, deux objets ont été prévus : « Preuve » et « Résultat ». Le premier sert lors de l'analyse de l'agent mobile et permet d'amasser une série de faits sur l'état du réseau. Le second permet de représenter l'information dans une structure utilisable par un gestionnaire. Dans notre cas, cette information est destinée à un gestionnaire humain. L'utilisation de ces structures est décrite en détail à la section 4.2.3.

Deux classes importantes redéfinissent les fonctions des agents fournis avec Grasshopper. Ils implémentent des méthodes précises pour retrouver les tables d'associations entre composantes et stations de gestion et pour communiquer avec les agents stationnaires implémentant le code de gestion. Ces classes se nomment « AgentMobileGestion » et « AgentGestion ». La classe « AgentMobileGestion » a aussi comme rôle de garder une référence au positionnement et au déplacement de l'agent mobile de gestion. Les deux classes permettent aussi de charger les habiletés de gestion en obtenant les interfaces de gestion appropriées. Les interfaces de gestion sont détaillées dans la section 4.1.2.

Les autres classes utilisées par l'architecture et les agents mobiles sont les classes fournies avec Java 1.3.1 et Grasshopper 2.2.4b. De plus, d'autres classes ont été fabriquées pour permettre une utilisation plus simple de l'architecture telles que des listes de classes implémentant des fonctions facilitant leur utilisation. Par exemple, une classe nommée « Chemin » permet d'ajouter plusieurs « StructureRoutage » et est utilisée par l'agent mobile de diagnostic pour conserver de l'information sur le chemin emprunté par une connexion entre deux ordinateurs.

### 4.1.2 Interfaces de gestion

Les interfaces de gestion sont les outils de gestion uniformes de l'architecture et permettent de gérer une composante sans nécessairement avoir connaissance de la technologie nécessaire pour cette gestion. L'agent mobile de gestion ne retient que la définition de cette interface (prototypes des fonctions) sans en connaître l'implémentation. Les classes « AgentMobileGestion » et « AgentGestion » fournissent les méthodes pour charger les bonnes interfaces de gestion et les bonnes technologies de gestion selon la composante courante gérée. Ces méthodes s'assurent que l'interface demandée est bel et bien implémentée par la technologie demandée. Par exemple, un premier agent stationnaire de gestion peut implémenter les interfaces de gestion de routage, d'interrogation et de performance, alors que le second agent stationnaire n'a pas les capacités technologiques pour implémenter l'interface de performance. Si la composante courante à gérer permet ces deux technologies et que l'agent mobile veut utiliser une interface de gestion de performance, l'architecture ne permettra l'utilisation que du premier agent stationnaire.

La Figure 4.3 montre le processus pour obtenir un proxy vers un agent stationnaire implémentant le code de gestion désiré. L'agent mobile doit interroger une composante nommée A. Il peut le faire via une interface nommée « Interrogation ». La composante A peut être gérée par les technologies B et C. La super-classe « AgentMobileGestion » tente alors de trouver un agent stationnaire répondant aux exigences de l'agent mobile.

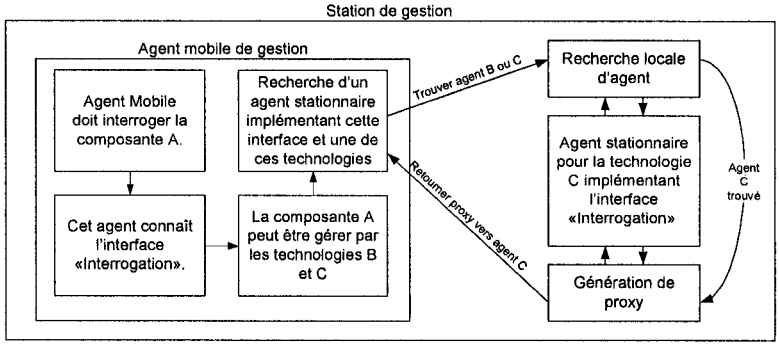


**Figure 4.3 Schéma démontrant l'utilisation du code technologique par proxy**

Elle commence donc par rechercher l'agent, puis demande ensuite, si la recherche est concluante, de créer un proxy vers cette composante. L'agent mobile est donc en mesure d'interroger la composante via l'interface de gestion. Ce scénario représente le cas où le chargement de code est dynamique. Dans le cas où on n'utilise que du code statique, l'agent mobile de gestion utilise l'interface, mais le code de gestion est aussi incorporé à l'agent mobile. Ce code n'est cependant pas déplacé avec l'agent mobile car il figure dans les éléments de base de l'architecture. En effet, toutes les classes se trouvant dans le « CLASSPATH » de la station de gestion sont chargées par Grasshopper et ne sont pas déplacées avec l'agent. Comme il a été mentionné, cette approche est plus rigide, mais plus performante. Ajoutons que bien que cela ne soit pas indiqué à la Figure 4.3, l'architecture tente de charger un agent stationnaire implémentant le code de gestion, s'il n'est pas déjà en exécution. S'il est impossible de charger le code par proxy, l'agent mobile doit être en mesure de continuer ou terminer sa tâche correctement en tenant compte de cette impossibilité.

Les interfaces de gestion prévues dans le cadre de notre architecture sont « IHabiletePerformance », « IHabileteInterrogation », « IHabileteRoutage », « IHabileteSNMP » et « IHabileteBase ». Ces cinq interfaces héritent de deux interfaces nommées « IHabilete » et « IProxyCheck ». La première définit une fonction permettant de vérifier si le système de gestion est fonctionnel, alors que la seconde permet de retourner le nom de la classe de l'agent stationnaire, ce qui est impossible à obtenir par l'objet « proxy » généré par Grasshopper. L'interface

« IHabiletePerformance » implémente des fonctions pour obtenir des mesures de performance. L'interface « IHabileteInterrogation » permet l'interrogation d'une composante. L'interface « IHabileteRoutage » regroupe toutes les habiletés ayant trait au routage telles que l'obtention d'une composante suivante. L'interface « IHabileteSNMP » permet la gestion SNMP d'une composante. Lorsque l'agent mobile utilise cette interface « IHabileteSNMP », il sait avec quelle technologie il désire gérer une composante. Ces quatre interfaces d'habiletés représentent les capacités de la composante. La cinquième interface, « IHabileteBase », représente ce qu'un agent mobile peut faire à partir de la station de gestion et n'est pas dépendante des technologies de gestion de la composante. Un aperçu des fonctions remplies par ces interfaces est donné au Tableau 4.1. Par exemple, une station de gestion peut s'exécuter sur une station Windows ou Linux et une composante peut utiliser comme moyen de gestion les protocoles SNMP et CMIP. Les quatre premières interfaces devraient avoir une implémentation SNMP et/ou CMIP. L'interface de base aurait une implémentation Windows et une Linux.

**Tableau 4.1 Description des fonctions fournies par les différentes interfaces**

| ***Habileté : Fonction de base héritée par les autres habiletés*** | |
|---|---|
| **Fonction** | **Description** |
| SystemeGestionDisponible | Indique si la composante peut être gérée avec cette interface et avec l'agent stationnaire trouvé |
| ***Habileté : Base*** | |
| **Fonctions** | **Description** |
| Ping | Effectue un « ping » |
| VerifierService | Permet de vérifier qu'un service ou un serveur est disponible |
| ***Habileté : Snmp*** | |
| **Fonctions** | **Description** |
| EnvoiRequete | Envoie une requête SNMP et obtient la réponse |
| EnvoiRequetes | Envoie plusieurs requêtes SNMP et obtient les réponses |
| ***Habileté : Performance*** | |
| **Fonctions** | **Description** |
| ObtenirCongestion | Donne le taux de congestion |
| ObtenirUtilisation | Donne le taux d'utilisation |
| ***Habileté : Routage*** | |
| **Fonctions** | **Description** |
| ComposantesSuivantes | Renvoie toutes les composantes suivantes |
| ComposanteSuivante | Renvoie la prochaine composante selon la destination |
| ***Habileté : Interrogation*** | |
| **Fonctions** | **Description** |
| ObtenirAdresseInterface | Renvoie les adresses d'une interface |
| ObtenirInformationInterface | Renvoie plusieurs informations sur une interface |
| ObtenirIndexInterface | Retourne le numéro d'une interface selon son adresse |
| ObtenirValeur | Retourne l'état d'une variable |
| ReponseComposante | Interroge la composante pour savoir si elle répond |

Les agents mobiles de gestion ne sont pas obligés d'utiliser ces interfaces. Nous définissons deux types d'agents mobiles, les agents généraux et les agents spécialisés. Les premiers utilisent généralement ces interfaces de gestion permettant une gestion globale et de haut niveau. Les seconds peuvent utiliser les interfaces de gestion et les moyens de gestion directement sans l'intermédiaire d'une interface.

### 4.1.3 Agents stationnaires de gestion

Nous avons présenté les interfaces de gestion dans la section précédente. Ces interfaces permettent à l'agent mobile de faire une gestion sans avoir la pleine connaissance des technologies de gestion de ces composantes. Dans cette section, nous présenterons les agents stationnaires qui implémentent le code pour utiliser les différentes technologies de gestion des composantes du réseau. Si un administrateur voulait que l'architecture prenne en compte une nouvelle technologie de gestion, c'est un tel module qu'il faudrait qu'il implémente. Les modules créés pour l'implémentation utilisent les technologies Windows, SNMP et Java. Le module implémentant l'interface de gestion de base « SBase » utilise le langage Java, mais aussi des langages natifs programmés pour Windows en C++. Pour gérer les composantes, deux modules technologiques sont disponibles : « SWindows » et « SSnmp ». Le premier utilise aussi une librairie programmée en code natif spécialement pour Windows. Le second utilise un « java beans » implémentant le protocole SNMP. Le module « SWindows » implémente les interfaces de gestion de routage, de performance et d'interrogation. Le module « SSnmp », implémente les mêmes interfaces, avec en plus, l'interface « IHabileteSnmp ». Les interfaces implémentées par chaque agent stationnaire de gestion sont indiquées au Tableau 4.2. Dans le cas où l'agent mobile n'utilise pas les mécanismes de « proxy » pour accéder à ces agents stationnaires, leur code de gestion doit être incorporé dans le code des agents mobiles.

**Tableau 4.2 Interfaces implémentées par les agents stationnaires de gestion**

| Agents stationnaires de gestion | Interfaces implémentées |
|---|---|
| SSnmp | Snmp<br>Interrogation<br>Performance<br>Routage |
| SBase | Base |
| SWindows | Interrogation<br>Performance<br>Routage |

### 4.1.4 Tables d'associations entre composantes et stations de gestion

Les tables d'associations entre composantes et stations de gestion sont des éléments devant nécessairement être dynamiques. À mesure que la taille du réseau grandit, cette table devient de plus en plus grande. Il est donc peu recommandable de laisser les agents mobiles transporter cette table. C'est donc le seul élément de l'architecture dont l'accès est obligatoirement par proxy. Cet accès se fait de façon analogue à l'obtention d'une habileté de gestion présentée à la section 4.1.2.

Cette table a en fait deux rôles : obtenir les stations de gestion liées à une composante et compléter l'information sur une composante à partir d'informations partielles. Par exemple, si on ne dispose que d'une adresse de réseau, la table peut fournir les autres adresses liées à cette composante. Cela peut s'avérer crucial lorsqu'on désire gérer une composante dont certaines interfaces sont en panne. La table est implémentée par un « hash map » qui utilise une composante comme clé de recherche. Deux composantes sont considérées égales si elles possèdent une même adresse de réseau. Cela pourrait causer quelques problèmes dans un réseau avec plusieurs sous-réseaux réutilisant les mêmes adresses et il faudrait, dans un tel cas, revoir la fonction d'égalité entre deux composantes. L'utilisation d'un « map » implique donc qu'une composante n'est gérée que par une station de gestion. Une station de gestion peut cependant gérer plus d'une composante. La table d'associations entre composantes et stations de gestion est incluse dans un agent stationnaire,

permettant l'emploi des mécanismes de « proxy » et de communications prévus par Grasshopper.

Pour éviter que l'architecture soit mise en échec en cas d'absence de cette table, la classe de gestion « AgentMobileGestion » a prévu la recherche de la table au sein d'une même région Grasshopper. Cette recherche pourrait aussi être étendue aux autres régions en utilisant l'information sur la dernière station visitée. Cette recherche rend donc l'architecture plus robuste, mais il est recommandé d'avoir une table par station de gestion. La Figure 4.4 démontre l'ordre de recherche d'un agent pour une table d'associations.

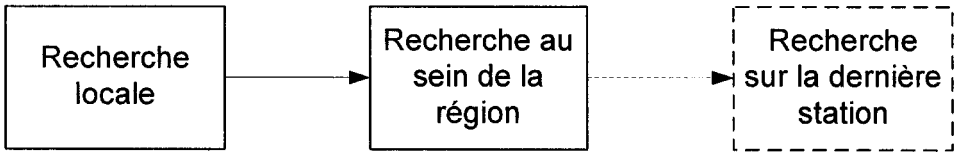


**Figure 4.4 Ordre de recherche d'une table d'associations**

La table d'associations doit comporter au minimum l'information pour les composantes avoisinantes du réseau. Cependant, cette limitation rendrait la gestion à distance difficile en cas de panne de stations de gestion. Il est donc recommandé que la table ait suffisamment d'informations sur le réseau pour pouvoir gérer des composantes qui sont éloignées. Le contenu des tables d'associations de différentes stations de gestion n'est pas tenu d'être identique.

### 4.1.5 Tolérance aux fautes du système de gestion

Une panne dans le système de gestion lui-même empêche le fonctionnement complet de l'agent mobile et de l'architecture. Ces derniers peuvent très bien fonctionner dans l'ensemble des composantes dont le système de gestion est intact. L'architecture est en mesure de notifier un problème dans le système de gestion et il suffit de réactiver l'élément déficient pour que l'architecture fonctionne complètement à nouveau. Cette détection est possible et est démontrée par l'agent mobile de

diagnostic qui est introduit à la section 4.2.3. Cependant, il ne peut pas à la fois trouver un problème dans le système de gestion et poursuivre son diagnostic pour trouver la cause du problème initial. Comme l'architecture essaie d'utiliser plusieurs technologies de gestion, il est peu probable, si l'administrateur a prévu plusieurs moyens de gestion, que l'agent mobile soit incapable de trouver une façon de gérer une composante. Bien entendu, si la composante ne fonctionne plus, les moyens de gestion, tout comme les fonctionnalités propres de la composante, deviennent inopérants. Ce cas est perçu comme une panne et non pas comme une panne du système de gestion.

## 4.2 Agents mobiles de gestion

Dans cette section, nous présentons les différents agents mobiles qui ont été créés pour le système de gestion de l'architecture. Les principaux agents sont : un agent mobile chargé de diagnostiquer des pannes simples dans un réseau à routage statique, et un agent mobile de recherche de chemin qui aide à préciser le diagnostic. Ces agents permettent au minimum de localiser la faute, et selon l'information disponible, ils peuvent à certains moments déterminer la cause de la panne. Nous verrons d'abord une série d'optimisations effectuées pour augmenter les performances de ces agents. Ensuite, il sera question des détails sur les différents agents mobiles.

### 4.2.1 Optimisation de la taille des agents mobiles de gestion

Pour éviter un maximum de transmissions lors de la migration des agents mobiles de gestion, un maximum de lignes de code a été installé sur chaque station de gestion. Cette pratique est motivée par le besoin de créer une architecture performante. Cependant, cela rend le système de gestion statique, étant donné que le code doit être fixé pour la durée de l'exécution des stations de gestion. Cette limitation est due à Grasshopper qui effectue une copie permanente de toutes les classes Java dans le « CLASSPATH ». Cependant, il est réaliste de croire que nous pourrions permettre la modification de ce code dynamiquement. Pour pallier ce problème, il est aussi possible de charger le code technologique de gestion par l'utilisation des communications par

proxy fournies par Grasshopper. Ainsi, le code n'est pas transporté par l'agent mobile et peut être modifié dynamiquement.

La construction de l'agent mobile de gestion et les choix de conception sont à la discrétion de l'administrateur du réseau. Ce dernier choisira la meilleure façon de procéder selon ses besoins. Pour permettre un maximum de performance, nous avons limité nos agents mobiles chargés du diagnostic à leur état et leurs données, en prenant soin de laisser le maximum de lignes de code sur les stations de gestion. Cela est possible en créant une classe dont la seule fonction est d'hériter de l'agent mobile de gestion. Cette approche permet de diminuer la taille du code transféré des agents mobiles, mais se fait au détriment de la simplicité de déploiement d'agents mobiles modifiés. Ce choix dépend de l'utilisation de l'architecture et de la possibilité d'arrêter le système de gestion.

L'agent mobile peut utiliser une variable pendant un certain nombre de sauts et ne plus l'utiliser quand son état change. Une optimisation possible consiste à affecter la valeur « null » aux variables Java qui ne seront plus utilisées. La machine virtuelle Java sait donc que ces variables peuvent être vidangées et évite de laisser l'agent mobile transporter de la mémoire qui ne sera plus utilisée.

### 4.2.2 Disposition des régions

Pour éviter certains problèmes liés aux partitionnements du réseau, plusieurs régions Grasshopper sont utilisées. Ce choix complique la recherche d'agents au sein d'autres régions, mais évite au maximum les problèmes liés aux agences n'ayant plus accès au registraire de la région. En effet, les agences gardent un lien avec la station de gestion ayant démarré le serveur de région et cela peut devenir problématique dans un environnement où les connexions sont peu fiables.

### 4.2.3 Fonctionnement de l'agent mobile de diagnostic

L'agent mobile de diagnostic est le principal agent qui a été développé dans le cadre de l'architecture proposée. C'est un agent mobile de gestion général, effectuant

un diagnostic avec des connaissances limitées des types de composantes du réseau. Cet agent mobile doit tout de même avoir quelques connaissances des types de composantes qu'il rencontre. La gestion est légèrement différente selon que l'agent interroge un routeur ou un commutateur. C'est en somme la seule spécialisation qu'il possède. Les algorithmes et schémas présentés dans cette section sont mieux détaillés que dans le chapitre précédent.

Cet agent mobile de gestion divise sa tâche en deux étapes principales. La première étape consiste en l'analyse physique de chaque composante. L'agent mobile est en mouvement pendant cette phase et accumule naïvement une série de faits en faisant sensiblement la même analyse d'une composante à une autre. Cette étape se termine lorsque l'agent mobile de diagnostic atteint sa destination ou lorsqu'il lui est impossible de poursuivre sa route. Pendant la seconde étape, l'agent mobile ne se déplace plus. Il effectue le diagnostic en utilisant les faits recueillis et en déduit une série de résultats. Il tente alors d'établir l'emplacement précis et la cause exacte du problème. Pour simplifier la représentation, nous allons expliquer le fonctionnement de l'agent mobile de diagnostic en plusieurs schémas distincts. Il faut ajouter que cet agent mobile ne prend en compte que les pannes simples et utilise les tables de routage pour se déplacer. Les routes sont statiques et demeurent les mêmes entre une source et une destination et vice-versa.

L'algorithme du parcours physique tient compte des limitations déjà énoncées ne permettant pas à toutes les composantes d'exécuter des agents mobiles. Une représentation schématique est présentée à la Figure 4.5. Les rectangles en gris représentent des étapes plus complexes dont les algorithmes sont représentés par la Figure 4.6 et la Figure 4.8. Le rectangle au centre nommé « Lancement de l'AMG RechercheChemin » représente une étape exécutée par un autre agent mobile de gestion qui est expliqué à la section 4.2.4. L'abréviation AMG désigne un agent mobile de gestion. Pour tous les schémas, les rectangles entourés de rectangles plus grands gris et blancs représentent respectivement des points d'entrée et de sortie.

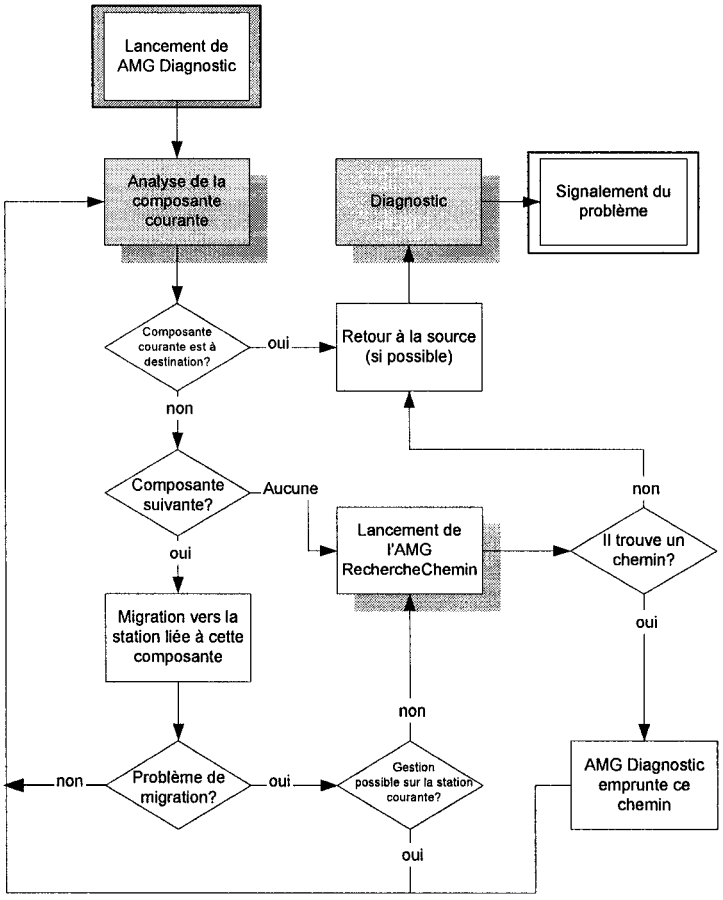


**Figure 4.5 Algorithme du parcours physique de l'agent mobile de diagnostic**

À la Figure 4.5, l'agent de diagnostic est lancé lorsqu'un problème de connexion est décelé entre une source et une destination. Il tente ensuite d'effectuer l'analyse de la composante courante (Figure 4.6) qui au départ est la source. Lorsque cette analyse est terminée, l'agent vérifie qu'il n'est pas arrivé à destination et obtient la composante suivante sur son chemin. L'agent mobile tente alors de migrer sur la station de gestion gérant cette composante. Si cette station de gestion est la même que la station courante, aucune migration n'est effectuée. Si l'agent ne peut trouver une composante suivante, il a l'option d'utiliser l'agent de recherche de chemin qui peut l'aider à trouver un chemin alternatif pour gérer la prochaine composante (voir section 4.2.4). Si une composante suivante existait et que la migration réussit, le processus recommence et l'agent analyse la nouvelle composante courante. Dans tous les autres cas, l'agent

mobile de diagnostic utilisera, en dernier recours, l'agent mobile de recherche de chemin. Si celui-ci trouve un chemin alternatif, l'agent mobile de diagnostic l'empruntera sans toutefois effectuer une analyse des composantes rencontrées sur ce chemin alternatif. En effet, ces composantes ne font pas partie du chemin original qu'aurait dû suivre la connexion en absence de problèmes. Si l'agent mobile de recherche de chemin échoue, ou si l'agent mobile de diagnostic n'utilise pas cette option, l'agent mobile de diagnostic tente alors de retourner vers la source pour effectuer son diagnostic. Cette dernière phase est expliquée plus loin (Figure 4.8). L'étape finale est le signalement du problème. Naturellement, si l'agent mobile réussit à atteindre la destination, il termine sa phase d'analyse et retourne vers la source pour effectuer son diagnostic.

L'analyse de la composante courante consiste à faire une série de tests qui serviront à la phase de diagnostic. Tous ces tests ne sont pas nécessairement concluants s'ils sont pris un à un. Cependant, lors de la phase de diagnostic, l'ensemble de ces tests peut permettre de poser un diagnostic précis. Cette précision est accrue lorsque toutes les composantes sur le chemin ont été analysées. Cette analyse améliorée est permise par l'agent de recherche de chemin. La Figure 4.6 montre les différentes étapes lors de l'étape « Analyse de la composante courante » de la Figure 4.5.

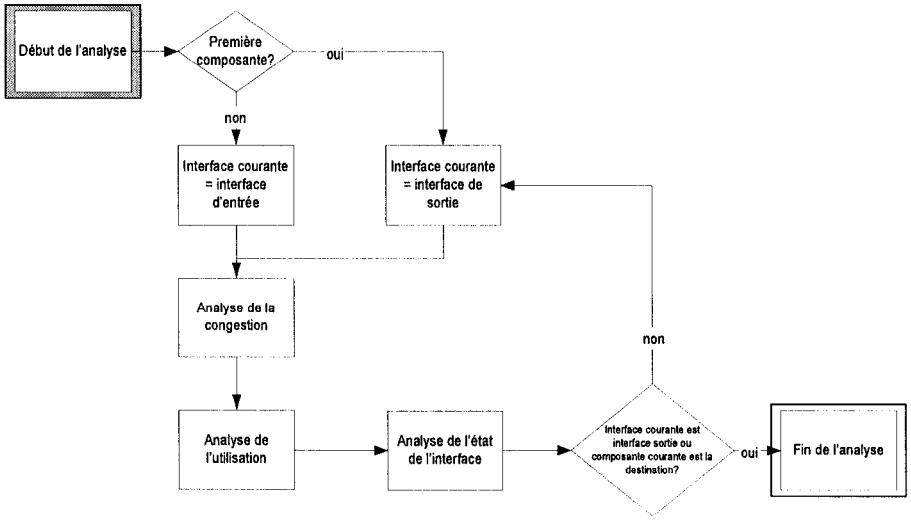


**Figure 4.6 Algorithme de l'analyse de la composante courante**

Cet algorithme se contente de faire une série de tests et d'accumuler des « éléments de preuve » dans une liste de preuves. Nous voyons que l'analyse s'effectue sur l'interface d'entrée et de sortie. Elle effectue deux analyses de performance et une analyse de l'état de l'interface. D'autres analyses pourraient se greffer à cet algorithme pour prendre en compte plus de possibilités d'erreurs. L'interface d'entrée n'est pas analysée lors du départ et l'interface de sortie n'est pas analysée lors de l'arrivée de l'agent mobile à la destination. Chaque interface examinée correspond à un point dans le chemin de l'agent mobile de diagnostic. La liste d'éléments de preuve est ajoutée à la fin de l'analyse de chacun de ces points. À la fin de son parcours physique, l'agent mobile de diagnostic aura donc en mémoire le chemin avec pour chaque point de ce chemin une liste de problèmes potentiels. Il ne faut pas confondre ce chemin avec le parcours de l'agent mobile Le chemin représente l'ordre dans lequel l'agent mobile a analysé les composantes à gérer. La Figure 4.7 donne une représentation du chemin.

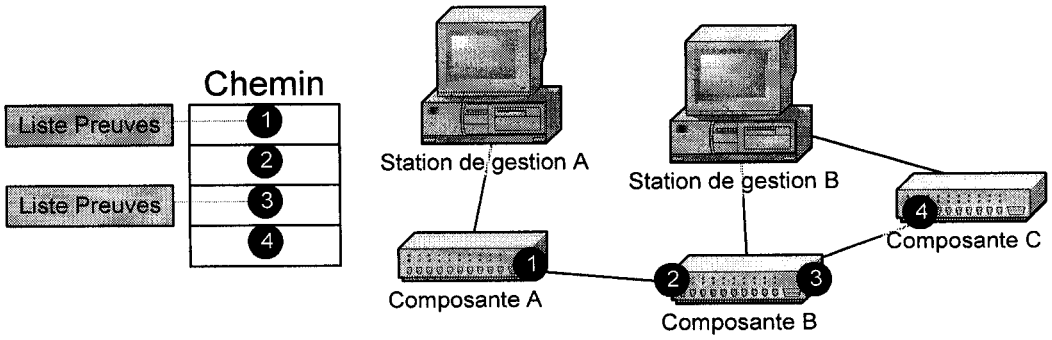


**Figure 4.7 Structure de l'information provenant de l'analyse**

À la Figure 4.7, le chemin est représenté par quatre éléments. Les boules noires représentent quatre points de ce chemin. Une liste de preuves est associée aux points de ce chemin où l'agent mobile de diagnostic a décelé des causes potentielles de la panne qui est à l'origine de son départ. C'est cette information qui est disponible à la dernière étape de l'agent mobile de diagnostic, qui est nommée « diagnostic ». Cette étape est représentée à la Figure 4.8.

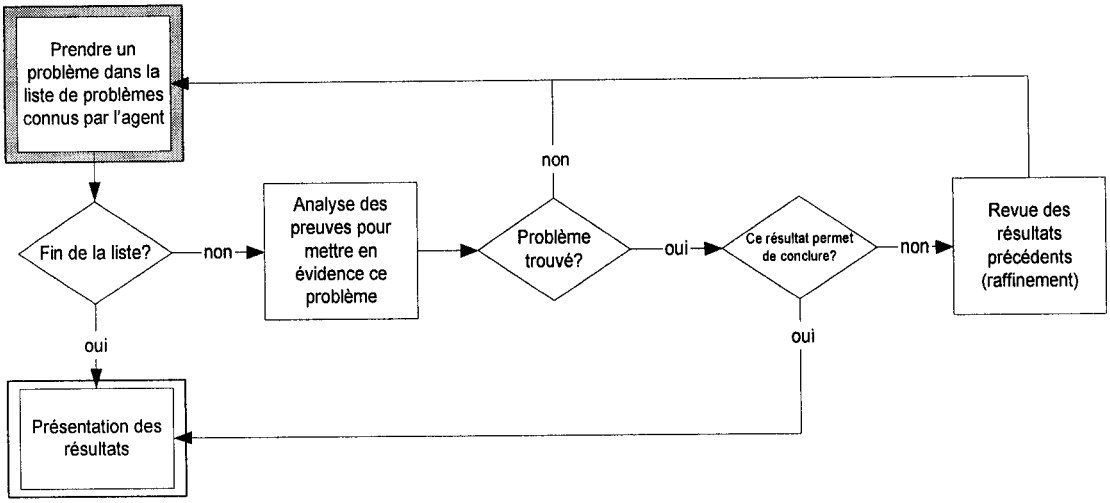


**Figure 4.8 Algorithme du diagnostic**

La phase de diagnostic utilise les preuves accumulées sur le parcours de l'agent mobile de diagnostic. Cette phase utilise un système expert de décision qui, par ordre de priorités, vérifie différents agencements de preuves pour en déterminer une cause précise. L'agent examine donc la liste des problèmes qu'il est capable de traiter. Selon les faits qu'il constate, il peut terminer son diagnostic s'il obtient un résultat satisfaisant. Sinon, il peut continuer de raffiner le résultat. Ainsi, un premier résultat pourrait être une interface qui s'est désactivée automatiquement. Le second résultat pourrait être la même conclusion, mais pour une autre interface. Avec le raffinement, l'agent mobile de diagnostic pourrait conclure que ces deux désactivations sont liées par un problème commun. La liste de problèmes à tester est constituée de manière intelligente. En effet, les problèmes vérifiés ne sont pas placés n'importe comment dans cette liste. Les tests pouvant conclure rapidement sont installés en premier. À toute étape, l'agent mobile peut soit terminer son exécution si le résultat est concluant, ou poursuivre le raffinement et la recherche de la solution.

La phase de diagnostic dépend donc de la phase d'analyse. En effet, l'analyse doit accumuler les éléments de preuve nécessaires au diagnostic. L'agent mobile de diagnostic définit trois types possibles de résultats : avertissements, intermédiaires et principaux. Cette classification est effectuée pendant le raffinement de la phase de

diagnostic. L'information disponible n'est pas toujours suffisante pour donner une cause précise d'erreur. Le classement en trois types a donc l'avantage de focaliser l'attention de l'administrateur sur les points principaux avant les intermédiaires ou même les avertissements. Ce dernier type est motivé par le fait qu'une panne détectée qui n'est pas la cause du mauvais fonctionnement initial peut tout de même être signalée. Le type de résultat principal tente de donner la cause exacte de la défaillance alors que les résultats intermédiaires servent soit à donner plus de précision sur le résultat principal ou à donner d'autres erreurs possibles qui pourraient être la cause du problème. Par exemple, si deux interfaces réseaux sont actives au point de vue administratif, mais sont inactives au point de vue opérationnel, cela peut signifier un problème de lien ou un problème dans une des deux interfaces. Le résultat principal indiquerait cette dernière conclusion, alors que les résultats intermédiaires indiqueraient les pannes sur les interfaces.

Il convient d'expliquer davantage les différentes classes créées pour l'agent mobile de diagnostic. La première se nomme « Preuve » et conserve un entier représentant un type de problème, un attribut pour qualifier ou quantifier ce problème et une référence à la composante où l'erreur est trouvée. Les preuves sont incorporées à une liste de preuves. Cette liste est ajoutée à chaque point du chemin dans une structure nommée « StructureRoutage » qui conserve les informations pertinentes sur l'interface analysée. Finalement, ces structures de routage sont ajoutées une à la suite de l'autre dans une liste nommée « Chemin ».

L'agent mobile présenté n'utilise pas de connaissances précises du réseau, mais si tel était le cas, le diagnostic pourrait être amélioré. Par exemple, notre réseau test utilise des interfaces ATM et Ethernet. Si l'agent mobile de diagnostic connaissait son environnement, il pourrait donner plus d'informations lorsqu'une panne survient sur une interface ATM. En effet, la configuration et la représentation des interfaces ATM ne sont pas les mêmes que pour Ethernet. Notre agent mobile donne cependant suffisamment d'informations pour bien cibler l'emplacement du problème.

### 4.2.4 Recherche de chemin

Cet agent mobile permet de préciser le diagnostic en tentant de trouver des routes alternatives pour atteindre la destination de l'agent mobile de diagnostic. Cependant, cette recherche a un certain coût, demandant l'envoi d'un agent sur tous les liens de communications sortant d'une composante. Cet agent mobile est activé par l'agent mobile de diagnostic lorsque celui-ci est incapable de déterminer ou d'atteindre la prochaine composante sur son chemin entre la source et la destination. Le succès de la recherche repose sur l'existence d'une autre route entre la source et la destination. À la Figure 4.9, le mécanisme de résolution entre composantes et stations de gestion n'est pas montré pour simplifier la représentation. La Figure 4.9 montre l'algorithme de l'agent mobile de recherche de chemin.

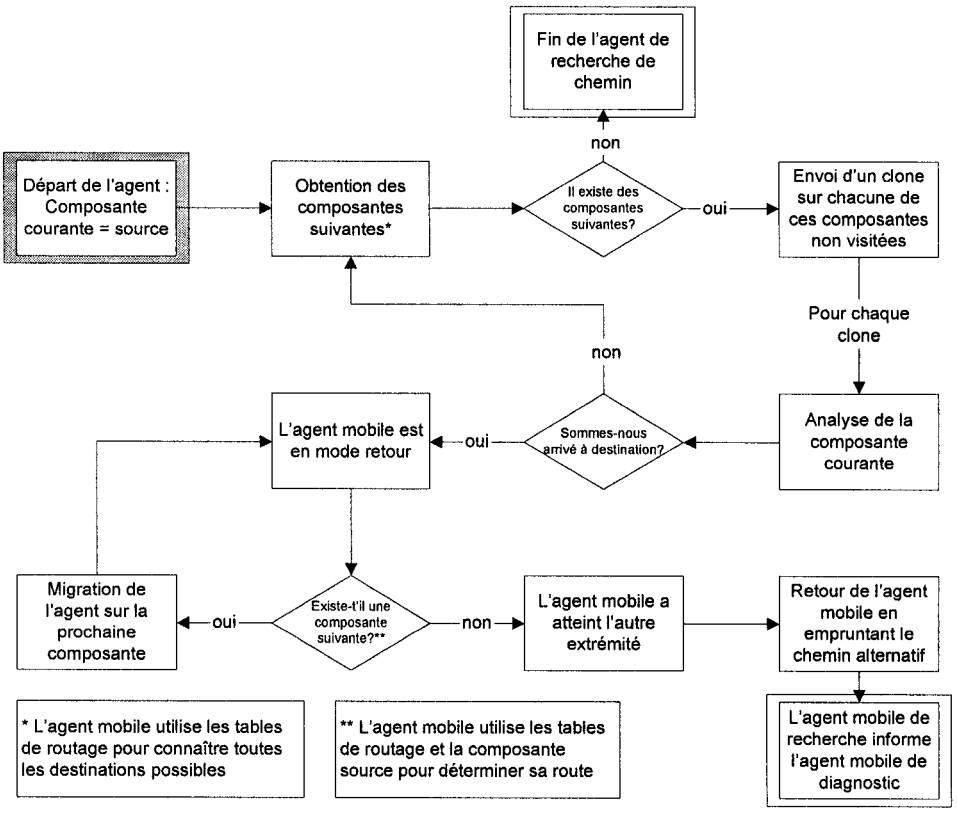


**Figure 4.9 Algorithme de l'agent mobile de recherche de chemin**

L'agent mobile de recherche essaie donc de retrouver la destination que l'agent mobile de diagnostic tente de joindre en se clonant sur toutes les composantes (en fait, toutes les stations de gestion liées à ces composantes) suivantes. Les clones se rappellent les composantes déjà visitées pour éviter une prolifération inutile d'agent mobile ou encore un mouvement circulaire. Les clones font une légère analyse de la composante courante pour vérifier si cette composante pourrait être la destination. Quand la destination est atteinte, l'agent mobile de recherche se met en mode retour et tente de retourner vers la source en utilisant les tables de routage. Lorsqu'il arrive à un point où la table de routage ne peut lui indiquer une route vers la source ou que la migration est impossible, l'agent mobile de recherche conclut donc qu'il a atteint la composante que l'agent mobile de diagnostic aurait dû atteindre. Il revient donc sur ses pas pour rejoindre et informer l'agent mobile de diagnostic de ce nouveau chemin. Ce dernier empruntera ce nouveau chemin sans effectuer d'analyse en cours de route. Il reprendra l'analyse sur la composante trouvée par l'agent mobile de recherche de chemin. La Figure 4.10 illustre en partie le résultat de l'interaction de ces deux agents.

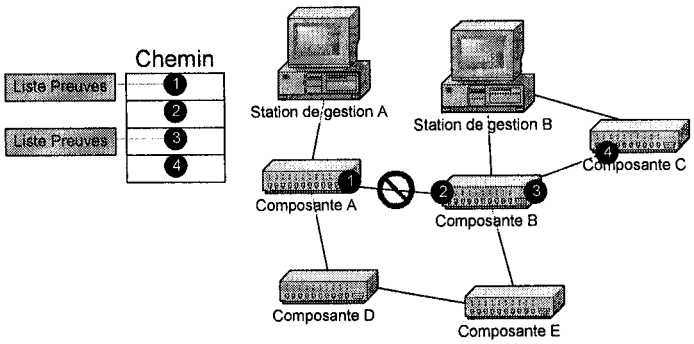


**Figure 4.10 Construction du chemin de l'agent mobile de diagnostic lorsqu'il est aidé par l'agent mobile de recherche de chemin**

Lorsque l'agent mobile de diagnostic se bute à l'impossibilité de migrer de la station de gestion A à la station de gestion B et qu'il ne peut pas non plus gérer la composante B, il lui est alors impossible de poursuivre son analyse sans l'aide de l'agent mobile de recherche de chemin. Il faut rappeler que nous utilisons des tables de routage statiques.

L'agent mobile de recherche trouve un chemin alternatif et se rend à la destination qui est, ici, la composante C. Arrivé à destination, il retourne en utilisant les tables de routage jusqu'à ce qu'il se trouve sur une composante où il lui est impossible de poursuivre son retour (composante B). Il réemprunte donc le chemin alternatif vers la source (composante A) pour informer l'agent mobile de diagnostic qui était bloqué. Celui-ci empruntera le chemin alternatif jusqu'à la composante B, et poursuivra son analyse. En réalité, ces composantes peuvent ne pas toujours être en mesure d'accueillir des agents mobiles. Dans ces cas, ce sont les stations de gestion qui accueillent les agents mobiles, mais le fonctionnement demeure le même. Nous verrons plus loin que cet agent mobile de recherche de chemin peut véritablement améliorer la précision et l'exactitude du diagnostic. Il doit avoir à sa disposition plusieurs stations de gestion pour effectuer une recherche efficace.

### 4.2.5 Mise à jour des tables d'associations

Un agent mobile de gestion permet la mise à jour des tables d'associations entre les diverses stations de gestion. Cet agent implémente un algorithme très simple qui consiste à lire la table d'associations locale et à la propager sur les autres stations de gestion connues.

Une variation de cet agent mobile pourrait être en mesure de découvrir lui-même les diverses composantes et ainsi effectuer la construction complète de cette table. Plus encore, il pourrait déterminer les associations optimales entre composantes et stations de gestion. En automatisant le processus de mise à jour, l'architecture deviendrait encore plus évolutive.

## 4.3 Tests et résultats

Le réseau choisi pour les tests utilise deux routeurs Ethernet/ATM, un routeur Ethernet et un commutateur ATM. Une station de gestion est liée à chaque routeur. Ces stations de gestion ont aussi comme rôle d'être des stations de travail. Comme cela a déjà été mentionné, pour simplifier l'algorithme de l'agent mobile de diagnostic, le

routage est statique. Les technologies de gestion utilisées sont : SNMP pour les routeurs et commutateurs, et Windows pour les stations de travail et de gestion. Les technologies de transport sont IP et ATM. Le Tableau 4.3 résume les composantes utilisées pour les tests.

**Tableau 4.3 Description des composantes du réseau test**

| Description | Possibilité de gestion | Système d'exploitation | Interfaces | Plateforme d'agents mobiles |
|---|---|---|---|---|
| Intel Pentium 4 1.7Ghz 256meg | oui | Windows 2000 SP3 | 1x100 Mbits | GrassHopper 2.2.4b JRE 1.3.1 |
| Intel Pentium 4 1.7Ghz 256meg | oui | Windows 2000 SP3 | 1x100 Mbits | GrassHopper 2.2.4b JRE 1.3.1 |
| Intel Pentium 4 1.7Ghz 256meg | oui | Windows 2000 SP3 | 1x100 Mbits | GrassHopper 2.2.4b JRE 1.3.1 |
| Routeur Cisco 3640 | oui | IOS 12.2 | 1x100 Mbits<br>1x10 Mbits<br>1xATM155Mbits | non applicable |
| Routeur Cisco 3640 | oui | IOS 12.1 | 1x100 Mbits<br>2x10 Mbits | non applicable |
| Routeur Cisco 3640 | oui | IOS 12.2 | 1x100 Mbits<br>1x10 Mbits<br>1xATM155Mbits | non applicable |
| Commutateur Cisco Catalyst 8500 | oui | IOS 12.0 | 2xATM 155Mbits | non applicable |
| Concentrateur 4 ports | non | non applicable | 4x100 Mbits | non applicable |

### 4.3.1 Installation des agents stationnaires

Les tests effectués utiliseront tous des agents mobiles qui accèdent au code technologique de gestion par proxy. Ces agents stationnaires sont donc installés sur chaque station avant les expériences. Ils résident en permanence sur les différentes stations de gestion. En prenant en considération qu'il n'y a pas de contraintes de mémoire sur les stations de gestion, nous avons installé les tables d'associations entre composantes et station de gestion sur chaque station de gestion pour éviter d'augmenter le trafic sur le réseau. En effet, si l'agent doit transporter cette information, il risque d'augmenter sa taille inutilement. Si nous n'installons cette information que sur certaines stations de gestion du réseau, un échec de communication peut paralyser le système de gestion en plus d'engendrer des communications inutiles. De plus, l'installation de ces tables sur chaque station permet une optimisation localisée. La Figure 4.11 illustre un réseau où il est possible d'optimiser la table d'associations entre composantes et stations de gestion. La composante centrale peut être gérée efficacement par deux stations de gestion.

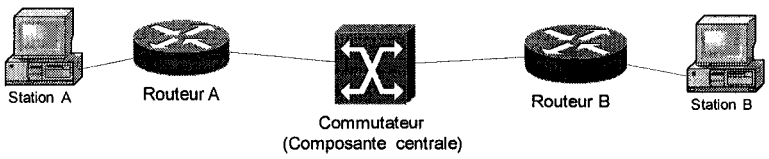


**Figure 4.11 Réseau induit avec commutateur central**

Il est évident que de faire migrer un agent mobile de la station A vers la station B pour gérer la composante centrale peut ne pas être avantageux. L'administrateur qui installe les tables peut optimiser la table de chacune des stations pour améliorer le choix de la station de gestion la plus près. Pour les tests, toutes les tables auront les mêmes associations.

## 4.3.2 Réseaux tests

Nous utiliserons deux réseaux tests pour vérifier deux données importantes. Le premier réseau test est construit avec trois routeurs et un commutateur. Il est représenté à la Figure 4.12.

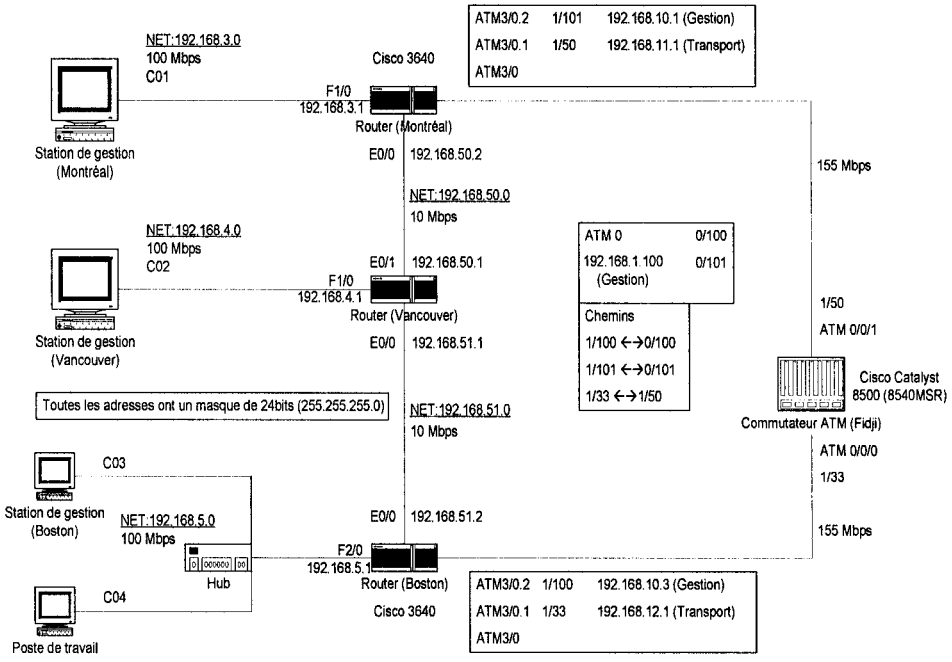


**Figure 4.12 Réseau en triangle pour les tests de diagnostic**

Ce réseau servira à démontrer la capacité de nos agents mobiles à effectuer une gestion efficace d'un vrai réseau de télécommunications. Le second réseau test, montré à la Figure 4.13, servira à vérifier les performances des agents mobiles en comparaison avec un agent stationnaire de gestion à distance. Les informations ne sont pas répétées à la Figure 4.13, étant donné que la configuration reste la même qu'à la Figure 4.12 à l'exception de la portion ATM du premier réseau qui est retirée. Ce dernier réseau utilise davantage la mobilité permettant de meilleures analyses de performance. Le premier réseau a la particularité d'avoir plusieurs routes, exploitant mieux l'agent mobile de recherche.

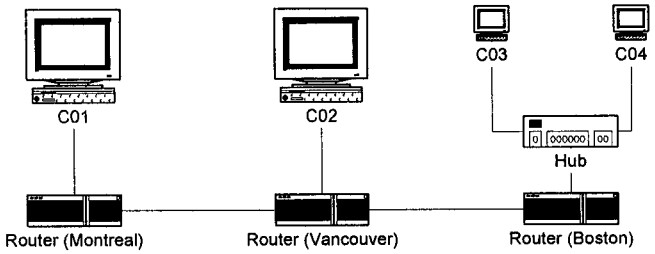


**Figure 4.13 Réseau linéaire pour les tests de diagnostic et de performance**

Il se prête moins bien à l'analyse de performance, car jamais plus d'un saut n'est nécessaire pour atteindre la station de gestion liée à une composante, peu importe la route.

La table d'associations entre composantes et stations de gestion installée sur chaque station de gestion est remplie comme au Tableau 4.4. Pour le réseau test linéaire, l'entrée avec le commutateur Fidji est retirée.

**Tableau 4.4 Table d'associations utilisée pour les tests**

| Composante | Station de gestion |
|---|---|
| Routeur Montréal | Montréal |
| Routeur Vancouver | Vancouver |
| Routeur Boston | Boston |
| Commutateur Fidji | Montréal |
| Station de gestion Montréal | Montréal |
| Station de gestion Vancouver | Vancouver |
| Station de gestion Boston | Boston |

### 4.3.3 Configuration des routeurs et du commutateur

Pour les besoins de l'expérience, les tables de routage sont statiques et ne définissent aucune route par défaut. Un problème d'interface ou de lien rend donc l'acheminement du trafic impossible vers certaines destinations. Ces choix ont été faits pour limiter la taille des problèmes que l'agent mobile de diagnostic devait envisager. Pour les besoins de gestion, le commutateur ATM a été configuré de telle sorte qu'il puisse répondre à des requêtes IP sur ses interfaces ATM. Il était donc nécessaire de configurer une table de routage statique pour le commutateur. Les tables de routage sur les routeurs ont été configurées de telle sorte que le trafic circulant de Montréal vers Boston et vice-versa passe par le commutateur ATM. Le trafic impliquant Vancouver comme source ou destination passe par le réseau Ethernet 10 Mbps. Les tables de routage complètes sont disponibles en annexes dans les configurations des routeurs.

### 4.3.4 Présentation des tests

Tous les tests présentent divers scénarios de pannes simples. Ils ont cependant tous un objectif différent à tester. Les trois premiers tests démontrent la capacité des différents agents à diagnostiquer les pannes simples dans le réseau test en triangle. Plus précisément, ces tests permettent de démontrer les capacités accrues de diagnostic du duo composé des agents mobiles de diagnostic et de recherche de chemin. Le quatrième test permet de démontrer le comportement de ce duo dans certains cas particuliers. Par exemple, nous voulions démontrer que les agents mobiles chargés du diagnostic et de la recherche ne trouve aucune panne lorsqu'il n'en a aucune et qu'ils peuvent au minimum nous indiquer si un problème survient au niveau du système de gestion. Les deux derniers tests comparent nez à nez l'agent mobile de diagnostic seul avec un agent stationnaire de diagnostic simulant un gestionnaire à distance. Pour vérifier des cas où les agents mobiles sont véritablement utilisés, nous avons proposé des scénarios où l'agent mobile de diagnostic doit migrer au moins une fois. Pour ce faire, nous avons utilisé le réseau test linéaire et nous n'avons programmé que des défaillances ayant lieu sur le routeur ou la station Boston. Les fiches techniques des

différents tests sont présentées au Tableau 4.5. Les différentes pannes simulées pour ces tests seront présentées avec les résultats.

**Tableau 4.5 Présentation des conditions des six tests**

| | | | |
|---|---|---|---|
| **Test #1** | *Démonstration de la précision des agents mobiles de diagnostic et de recherche* | | |
| Routage: | statique | Recherche de chemins alternatifs: | oui |
| Modèle de panne: | simple | Modèle de mobilité: | proximité |
| Code des agents mobiles: | jar + proxy | Tables composantes-stations: | inst. sur ch. stations |
| Réseau test | triangle | | |
| **Test #2** | *Même configuration que le test #1, mais avec l'agent mobile de diagnostic seulement* | | |
| Routage: | statique | Recherche de chemins alternatifs: | non |
| Modèle de panne: | simple | Modèle de mobilité: | proximité |
| Code des agents mobiles: | jar + proxy | Tables composantes-stations: | inst. sur ch. stations |
| Réseau test | triangle | | |
| **Test #3** | *Même configuration que le test #1, mais avec un agent stationnaire de diagnostic seulement* | | |
| Routage: | statique | Recherche de chemins alternatifs: | non |
| Modèle de panne: | simple | Modèle de mobilité: | aucune mobilité |
| Code des agents mobiles: | jar + proxy | Tables composantes-stations: | inst. sur ch. stations |
| Réseau test | triangle | | |
| **Test #4** | *Cas spéciaux : aucune pannes et pannes de gestion* | | |
| Routage: | statique | Recherche de chemins alternatifs: | oui |
| Modèle de panne: | simple | Modèle de mobilité: | proximité |
| Code des agents mobiles: | jar + proxy | Tables composantes-stations: | inst. sur ch. stations |
| Réseau test | triangle | | |
| **Test #5** | *Utilisation de l'agent mobile de diagnostic seulement* | | |
| Routage: | statique | Recherche de chemins alternatifs: | non |
| Modèle de panne: | simple | Modèle de mobilité: | proximité |
| Code des agents mobiles: | jar + proxy | Tables composantes-stations: | inst. sur ch. stations |
| Réseau test | linéaire | | |
| **Test #6** | *Utilisation de l'agent stationnaire de diagnostic seulement* | | |
| Routage: | statique | Recherche de chemins alternatifs: | non |
| Modèle de panne: | simple | Modèle de mobilité: | aucune mobilité |
| Code des agents mobiles: | jar + proxy | Tables composantes-stations: | inst. sur ch. stations |
| Réseau test | linéaire | | |

### 4.3.5 Résultats et analyse

Dans cette sous-section, nous présentons tous les résultats obtenus. Les premiers tableaux (tableaux 4.6, 4.7 et 4.8), à la page suivante, représentent les expériences faites sur le réseau triangle dans trois cas différents : utilisation des agents mobiles de diagnostic et de recherche, utilisation de l'agent mobile de diagnostic seulement, et utilisation de l'agent stationnaire de diagnostic seulement. Les valeurs en gras représentent les meilleurs résultats. Les pannes simulées sont les mêmes pour les trois tests. Ces tests donnent deux mesures principales : le temps de réponse et l'exactitude du diagnostic. L'exactitude se définit à trois niveaux. Le premier niveau nommé « cause précisée » indique que l'agent a pu donner la cause exacte comme résultat principal et unique. Le second niveau, « cause identifiée », indique que la cause a bel et bien été identifiée, mais fait partie d'une série de suggestions où aucune cause n'est prédominante. Le dernier niveau, « cause voisine », indique que l'agent de diagnostic a indiqué une cause touchant une composante voisine à la réelle cause de la panne. Le premier test (Tableau 4.6) démontre que l'utilisation d'un agent mobile recherchant des chemins alternatifs augmente de beaucoup la précision du diagnostic. En effet, par rapport au Tableau 4.7 et au Tableau 4.8, les causes sont plus souvent précisées et elles sont toujours au minimum identifiées. Cependant, cela se fait toujours au détriment du temps de réponse. Nous attribuons le temps de réponse plus élevé caractéristique des tests avec agents mobiles aux délais imposés par les attentes des délais de garde (timeouts) de migration. De plus, cette même migration prend plus de temps globalement que le temps utilisé pour la gestion à distance. Finalement, dans le cas du duo d'agents mobiles, le temps de réponse est encore plus long, compte tenu de l'attente de la fin de recherche de l'agent mobile de recherche. Pour permettre une comparaison honnête, nous avons obligé les agents mobiles à revenir à la destination avec leur diagnostic, question de donner le diagnostic sur la machine ayant démarré l'agent mobile. Ainsi, la comparaison peut être faite directement avec la gestion à distance. Les diagnostics posés par l'agent de diagnostic mobile et stationnaire, sans l'assistance de l'agent mobile de recherche, sont identiques.

## Tableau 4.6 Résultats pour les agents mobiles de diagnostic et de recherche de chemin (test #1)

| Session | Station de gestion Source | Destination | Cause de la panne | Cause précisée | identifiée | voisine | Temps de réponse (s) |
|---|---|---|---|---|---|---|---|
| 01 | Montréal | Boston | Lien ATM0/0/1 | X | X | | 70,87 |
| 02 | Montréal | Boston | Lien ATM0/0/0 | X | X | | 66,74 |
| 03 | Montréal | Boston | Interface ATM0/0/1 (admin down) sur Fidji | X | X | | 68,89 |
| 04 | Montréal | Boston | Interface ATM0/0/0 (admin down) sur Fidji | x | x | | 69,06 |
| 05 | Montréal | Boston | Interface ATM3/0.1 (admin down) sur routeur Boston | X | X | | 60,80 |
| 06 | Montréal | Boston | Lien Station Boston vers Hub | | x | | 85,87 |
| 07 | Montréal | Boston | Interface ATM3/0.1 (admin down) sur routeur Montréal | x | x | | 67,97 |
| 08 | Montréal | Vancouver | Lien Routeur Vancouver vers Montréal | X | X | | 26,94 |
| 09 | Montréal | Vancouver | Interface E0/0 (admin down) sur routeur Montréal | x | x | | 50,30 |
| 10 | Montréal | Vancouver | Interface E0/1 (admin down) sur routeur Vancouver | X | X | | 67,32 |
| 11 | Montréal | Vancouver | Crash du routeur Vancouver | | x | | 14,73 |
| 12 | Montréal | Vancouver | Service non démarré sur station Vancouver | x | x | | 9,37 |
| 13 | Montréal | Vancouver | Crash de la station Vancouver | | x | | 141,18 |
| 14 | Montréal | Vancouver | Lien Station Vancouver vers Vancouver | | x | | 69,45 |
| 15 | Vancouver | Montréal | Interface E0/0 (admin down) sur routeur Montréal | X | X | | 70,03 |
| 16 | Vancouver | Montréal | Crash sur routeur Montréal | | x | | 51,45 |
| 17 | Vancouver | Boston | Interface E0/0 (admin down) sur routeur Vancouver | x | x | | 55,62 |
| 18 | Vancouver | Boston | Interface F2/0 (admin down) sur routeur Boston | x | x | | 96,29 |
| 19 | Boston | Vancouver | Interface E0/0 (admin down) sur routeur Boston | x | x | | 57,65 |
| 20 | Boston | Montréal | Lien Station Montréal vers Routeur | | x | | 87,56 |

## Tableau 4.7 Résultats pour l'agent mobile de diagnostic (test #2)

| Session | Station de gestion Source | Destination | Cause de la panne | Cause précisée | identifiée | voisine | Temps de réponse (s) |
|---|---|---|---|---|---|---|---|
| 01 | Montréal | Boston | Lien ATM0/0/1 | | x | | **2,26** |
| 02 | Montréal | Boston | Lien ATM0/0/0 | | x | | 63,45 |
| 03 | Montréal | Boston | Interface ATM0/0/1 (admin down) sur Fidji | | x | | 3,13 |
| 04 | Montréal | Boston | Interface ATM0/0/0 (admin down) sur Fidji | x | x | | 64,09 |
| 05 | Montréal | Boston | Interface ATM3/0.1 (admin down) sur routeur Boston | | | x | 63,48 |
| 06 | Montréal | Boston | Lien Station Boston vers Hub | | x | | 37,97 |
| 07 | Montréal | Boston | Interface ATM3/0.1 (admin down) sur routeur Montréal | x | x | | 3,61 |
| 08 | Montréal | Vancouver | Lien Routeur Vancouver vers Montréal | | x | | 3,45 |
| 09 | Montréal | Vancouver | Interface E0/0 (admin down) sur routeur Montréal | x | x | | 2,41 |
| 10 | Montréal | Vancouver | Interface E0/1 (admin down) sur routeur Vancouver | | | x | 61,69 |
| 11 | Montréal | Vancouver | Crash du routeur Vancouver | | x | | **2,28** |
| 12 | Montréal | Vancouver | Service non démarré sur station Vancouver | x | x | | 7,52 |
| 13 | Montréal | Vancouver | Crash de la station Vancouver | | x | | 39,23 |
| 14 | Montréal | Vancouver | Lien Station Vancouver vers Vancouver | | x | | 29,80 |
| 15 | Vancouver | Montréal | Interface E0/0 (admin down) sur routeur Montréal | | x | | 62,30 |
| 16 | Vancouver | Montréal | Crash sur routeur Montréal | | x | | 67,21 |
| 17 | Vancouver | Boston | Interface E0/0 (admin down) sur routeur Vancouver | x | x | | 5,34 |
| 18 | Vancouver | Boston | Interface F2/0 (admin down) sur routeur Boston | x | x | | 30,07 |
| 19 | Boston | Vancouver | Interface E0/0 (admin down) sur routeur Boston | x | x | | 2,66 |
| 20 | Boston | Montréal | Lien Station Montréal vers Routeur | | x | | 28,88 |

## Tableau 4.8 Résultats pour l'agent stationnaire de diagnostic (test #3)

| Session | Station de gestion Source | Destination | Cause de la panne | Cause précisée | identifiée | voisine | Temps de réponse (s) |
|---|---|---|---|---|---|---|---|
| 01 | Montréal | Boston | Lien ATM0/0/1 | | x | | 3,26 |
| 02 | Montréal | Boston | Lien ATM0/0/0 | | x | | **41,17** |
| 03 | Montréal | Boston | Interface ATM0/0/1 (admin down) sur Fidji | | x | | **2,14** |
| 04 | Montréal | Boston | Interface ATM0/0/0 (admin down) sur Fidji | x | x | | **40,62** |
| 05 | Montréal | Boston | Interface ATM3/0.1 (admin down) sur routeur Boston | | | x | **40,46** |
| 06 | Montréal | Boston | Lien Station Boston vers Hub | | x | | **18,84** |
| 07 | Montréal | Boston | Interface ATM3/0.1 (admin down) sur routeur Montréal | x | x | | **2,36** |
| 08 | Montréal | Vancouver | Lien Routeur Vancouver vers Montréal | | x | | **2,38** |
| 09 | Montréal | Vancouver | Interface E0/0 (admin down) sur routeur Montréal | x | x | | **2,09** |
| 10 | Montréal | Vancouver | Interface E0/1 (admin down) sur routeur Vancouver | | | x | **39,11** |
| 11 | Montréal | Vancouver | Crash du routeur Vancouver | | x | | 3,16 |
| 12 | Montréal | Vancouver | Service non démarré sur station Vancouver | x | x | | **6,96** |
| 13 | Montréal | Vancouver | Crash de la station Vancouver | | x | | **14,58** |
| 14 | Montréal | Vancouver | Lien Station Vancouver vers Vancouver | | x | | **7,22** |
| 15 | Vancouver | Montréal | Interface E0/0 (admin down) sur routeur Montréal | | x | | **39,94** |
| 16 | Vancouver | Montréal | Crash sur routeur Montréal | | x | | **40,65** |
| 17 | Vancouver | Boston | Interface E0/0 (admin down) sur routeur Vancouver | x | x | | **2,06** |
| 18 | Vancouver | Boston | Interface F2/0 (admin down) sur routeur Boston | x | x | | **6,02** |
| 19 | Boston | Vancouver | Interface E0/0 (admin down) sur routeur Boston | x | x | | **2,44** |
| 20 | Boston | Montréal | Lien Station Montréal vers Routeur | | x | | **6,09** |

Le prochain test, dont les résultats sont au Tableau 4.9, démontre que les agents mobiles de diagnostic et de recherche de chemin sont en mesure de déceler une panne au niveau du système de gestion et qu'ils ne décèlent aucune panne lorsqu'il n'y en a aucune.

**Tableau 4.9 Résultats pour les cas particuliers**

| Session | Station de gestion | | Composante en panne | Cause de la panne | Temps de réponse (s) | Résultats |
|---|---|---|---|---|---|---|
| | Source | Destination | | | | |
| 01 | Montréal | Boston | Aucune | Aucune | 12.09 | Aucune erreur |
| 02 | Montréal | Vancouver | Aucune | Aucune | 7.86 | Aucune erreur |
| 03 | Vancouver | Montréal | Aucune | Aucune | 12.40 | Aucune erreur |
| 04 | Vancouver | Boston | Aucune | Aucune | 6.69 | Aucune erreur |
| 05 | Boston | Montréal | Aucune | Aucune | 10.02 | Aucune erreur |
| 06 | Boston | Vancouver | Aucune | Aucune | 8.34 | Aucune erreur |
| 07 | Montréal | Boston | Commutateur Fidji | Gestion désactivée | 189.24 | Erreur decelée |
| 08 | Montréal | Vancouver | Routeur Montréal | Gestion désactivée | 194.02 | Erreur decelée |
| 09 | Vancouver | Boston | Routeur Boston | Gestion désactivée | 195.08 | Erreur decelée |
| 10 | Boston | Montréal | Routeur Boston | Gestion désactivée | 195.56 | Erreur decelée |
| 11 | Boston | Vancouver | Routeur Vancouver | Gestion désactivée | 203.43 | Erreur decelée |

Dans les cas où une station de gestion était en panne, l'agent mobile de diagnostic était en mesure d'identifier la composante dont le système de gestion était en panne. Pour qu'un système de gestion soit considéré en panne, il faut que tous les moyens de gestion échouent. C'est le cas si le service SNMP n'est pas démarré, si une librairie importante de gestion est manquante et que le code technologique de gestion n'est pas disponible. Les temps de réponse élevés s'expliquent par l'attente des agents mobiles de recherche qui ne retournent pas. Ces temps de réponse pourraient être améliorés si les agents mobiles de recherche avaient un moyen d'indiquer à l'agent mobile de diagnostic que tous les agents mobiles de recherche créés n'ont pu trouver de chemin. Il faut noter qu'une panne dans le système de gestion ne garantit plus l'obtention d'un diagnostic.

Les deux derniers tests ont pour but l'évaluation de la performance de l'agent de diagnostic, sans l'aide de l'agent de recherche de chemin. L'agent de diagnostic est évalué avec mobilité et sans mobilité. Une topologie linéaire a été choisie pour les raisons évoquées à la section 4.3.4. Le Tableau 4.10 résume les résultats communs des deux tests. La source est toujours la station Montréal et la destination est toujours la

station Boston. Les pannes touchent exclusivement le routeur et la station Boston. Le diagnostic obtenu par l'agent de diagnostic est donc identique autant dans sa version mobile que stationnaire. Les résultats du Tableau 4.7 et du Tableau 4.8 démontrent davantage de simulations de pannes qui appuient cette affirmation.

**Tableau 4.10 Résultats communs pour les agents de diagnostic mobile et stationnaire (test #5 et #6)**

| Session | Composante en panne | Cause de la panne | Cause | | |
|---|---|---|---|---|---|
| | | | précisée | identifiée | voisine |
| 01 | Lien Station Boston vers Hub | Lien Station Boston vers Hub | | x | |
| 02 | Routeur Boston | Crash | | x | |
| 03 | Station Boston | Service non démarré | x | x | |
| 04 | Routeur Boston | Interface F2/0 (admin down) | | x | |
| 05 | Routeur Boston | Interface E0/0 (admin down) | | | x |
| 06 | Aucune | Aucune | x | x | |

Comme l'agent mobile de diagnostic est conçu pour une gestion proche, mais possiblement distante, les résultats identiques sont donc tout à fait normaux. L'agent de diagnostic stationnaire n'est qu'une version de l'agent mobile qui ne peut tout simplement pas se déplacer.

**Tableau 4.11 Résultats de performance pour l'agent de diagnostic mobile (test #5)**

| Session | Trafic | | | | Temps de réponse (s) |
|---|---|---|---|---|---|
| | Montreal | Vancouver | Boston | Total | |
| 01 | **46092** | 74988 | 12062 | 133142 | 47,18 |
| 02 | 43714 | 49876 | 0 | 93590 | 10,13 |
| 03 | **43206** | 59861 | 42380 | 145447 | 16,89 |
| 04 | **52936** | 113634 | 27510 | 194080 | 32,90 |
| 05 | 46541 | 46540 | 1342 | 94423 | 65,39 |
| 06 | **41099** | 59339 | 41469 | 141907 | 11,24 |

**Tableau 4.12 Résultats de performance pour l'agent de diagnostic stationnaire (test #6)**

| Session | Trafic | | | | Temps de réponse (s) |
|---|---|---|---|---|---|
| | Montreal | Vancouver | Boston | Total | |
| 01 | 62209 | **29051** | **11642** | **102902** | **15,62** |
| 02 | **41895** | **12313** | 0 | **54208** | **4,06** |
| 03 | 70328 | **35707** | **22589** | **128624** | **9,72** |
| 04 | 104449 | **59042** | **27120** | **190611** | **7,39** |
| 05 | **35584** | **10326** | **703** | **46613** | **39,03** |
| 06 | 70775 | **36408** | **23988** | **131171** | **6,65** |

Contrairement aux tests #1 et #2 (tableaux 4.6 et 4.7), les tests #5 et #6 (tableaux 4.11 et 4.12) utilise une topologie qui permet à l'agent mobile d'effectuer plus de sauts. Ces tests visaient à mettre en évidence des scénarios où l'agent mobile pourrait surpasser l'agent stationnaire de diagnostic, qui incarne le modèle de gestion à distance. Les mesures de temps de réponse et de trafic pour les deux derniers tests sont indiquées aux tableaux 4.11 et 4.12. Les nombres en gras indiquent les meilleurs résultats d'un test par rapport à l'autre dans leurs colonnes respectives. La prépondérance de l'agent stationnaire est indéniable. Il génère toujours moins de trafic au total et obtient toujours de meilleurs temps de réponses. Cependant, l'agent mobile tend à utiliser moins le premier routeur (routeur Montréal) sur sa route. En effet, la majorité de la gestion est effectuée sur la station Vancouver, qui est toujours considérée plus proche du routeur Vancouver. De plus, quatre des six sessions utilisent une panne empêchant le déplacement de l'agent de la station Vancouver à Boston. Cela explique que pour le cas de l'agent mobile, les mesures de trafic sont plus élevées pour le routeur Vancouver.

Les mesures de trafic ont été prises en effectuant une lecture SNMP sur chacun des routeurs à partir des stations de gestion liées directement. La différence du nombre d'octets en sortie a été utilisée. Pour mesurer le trafic sur les LAN (Local Area Network), nous avons utilisé le nombre d'octets en entrée sur les interfaces des routeurs connectant les trois LAN. Le biais introduit par ces lectures est infime. Il est d'environ 300 octets par routeur pour un total de 900 octets. Cette valeur n'a pas été soustraite du total. Il faut aussi noter que les routeurs échangent de l'information entre eux à raison de 60 octets par interface par dix secondes. Ce trafic n'a pas été soustrait non plus, faisant partie de la gestion. Ces 60 octets sont attribuables à la fonction « keep alive » des routeurs Cisco. Au Tableau 4.13, nous voyons les écarts dans les mesures de trafic de l'agent mobile vis-à-vis l'agent stationnaire.

**Tableau 4.13 Écarts entre le trafic de l'agent mobile et celui de l'agent stationnaire**

| Session | Trafics totaux | | Écart Trafic |
|---|---|---|---|
| | Mobile | Stationnaire | |
| 01 | 133142 | 102902 | 29,39% |
| 02 | 93590 | 54208 | 72,65% |
| 03 | 145447 | 128624 | 13,08% |
| 04 | 194080 | 190611 | 1,82% |
| 05 | 94423 | 46613 | 102,57% |
| 06 | 141907 | 131171 | 8,18% |

À la session 4, le trafic de gestion est plus élevé qu'ordinaire. Dans ce cas précis, l'agent stationnaire et l'agent mobile sont presque vis-à-vis en terme de trafic généré. Comme le trafic de gestion est très élevé, le surplus de trafic causé par la mobilité de l'agent mobile devient moins important. Il convient aussi d'analyser le trafic moyen total et pour chacun des routeurs. Le Tableau 4.14 donne ces valeurs moyennes et la Figure 4.14 donne une représentation graphique de ces moyennes.

**Tableau 4.14 Trafic moyen total et pour chacun des routeurs (test #5 et test #6)**

| | Trafic (octets) | | | |
|---|---|---|---|---|
| | Montréal | Vancouver | Boston | Total |
| Agent mobile | 45598 | 67373 | 20794 | 133765 |
| Agent stationnaire | 64207 | 30475 | 14340 | 109022 |
| Écart | 41% | -55% | -31% | -18% |

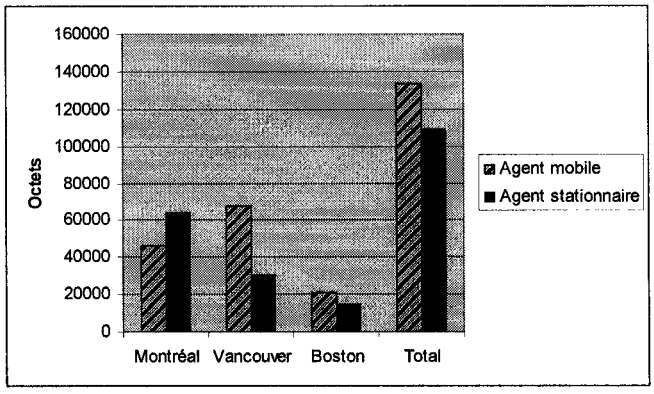


**Figure 4.14 Comparaison du trafic moyen engendré par les agents mobiles et stationnaires**

Il apparaît que le trafic sur le premier routeur est en moyenne 41% plus élevé pour l'agent stationnaire. C'est le second routeur qui profite le plus de la gestion à distance

avec 55% moins de trafic pour l'agent stationnaire. Au total, l'utilisation des agents mobiles ne semble pas augmenter considérablement le trafic global. Nous avons obtenu 18% moins de trafic avec l'agent stationnaire.

Les résultats indiquent à priori que d'utiliser l'agent de diagnostic seul n'est pas avantageux vis-à-vis la gestion à distance, autant en terme de trafic, de temps de réponse que de précision. Cependant, il faut préciser que notre algorithme de diagnostic pourrait utiliser davantage d'informations pour mener son diagnostic, ce qui pourrait modifier ces conclusions. Nous revenons davantage sur ces résultats dans la prochaine section.

### 4.3.6 Discussion des résultats

En observant les résultats, il est maintenant évident que les agents mobiles peuvent difficilement rivaliser sur tous les fronts avec la gestion à distance. Il n'en demeure pas moins qu'ils offrent certains avantages indéniables. Ils ont réellement tendance à distribuer la charge de manière constante sur le réseau. Ils diminuent l'impact de la gestion sur les premiers nœuds du réseau et offrent une capacité de gestion accrue en utilisant des outils locaux.

Les résultats laissent entrevoir que l'utilisation de plusieurs routeurs pourrait mener à des scénarios où l'agent mobile est plus performant, principalement si on combine l'effet de plusieurs agents ensemble. En effet, les résultats ont démontré que l'agent stationnaire de diagnostic tend à surutiliser le premier nœud du réseau, alors que l'agent mobile est plus constant sur cette utilisation. Tout porte donc à croire que les agents mobiles, dans un cas où plusieurs tâches sont effectuées en même temps, deviendraient plus performants. De plus, dans un cas précis, nous avons vu que l'agent mobile est en position de statu quo avec l'agent stationnaire. Dans ce cas, le trafic dû à la gestion, et non pas la mobilité, était très élevé. Le fait de gérer le problème à partir de la station Montréal (cas stationnaire) plutôt que la station Vancouver (cas mobile) a suffit à produire un cas où l'agent mobile semblait avantagé. Ce résultat laisse croire que si nous augmentions le nombre de preuves à collecter pour effectuer un diagnostic

encore plus précis et spécialisé, nous arriverions à démontrer que les agents mobiles offrent de meilleures performances que les approches de gestion à distance. Le but premier de l'architecture était de démontrer la possibilité de gérer un réseau avec des agents mobiles. Les expériences indiquent que cet objectif a été atteint. Du point de vue des performances, nos agents mobiles ont tout de même réussi à ne pas imposer une charge trop élevée au réseau. En effet, l'agent stationnaire génère seulement 18% moins de trafic en moyenne dans l'ensemble du réseau.

D'autres constatations plus qualitatives peuvent être apportées à la discussion. Les agents mobiles ont la capacité de continuer à fonctionner pendant un partitionnement. Ils ont démontré une application où la combinaison de deux agents arrivait à trouver, simplement, les causes de pannes plus précisément. Pour obtenir ce résultat avec une approche distante, il aurait fallu créer plusieurs connexions entre plusieurs routeurs. Bien que cela soit possible, il n'en demeure pas moins que les agents mobiles arrivent à cette précision d'une manière beaucoup plus naturelle et distribuée.

L'architecture s'est avérée bien adaptée à un réseau hétérogène. L'utilisation de plusieurs interfaces de gestion permet de séparer les moyens de gestion. Il devient donc possible d'utiliser de multiples technologies, même si certaines ne peuvent pas être utilisées pour toutes les tâches de gestion. Cette séparation pourrait donc s'avérer intéressante lorsque plusieurs technologies de gestion sont utilisées. L'architecture a démontré sa capacité à gérer tous les types de composantes, peu importe leur capacité ou leur technologie de gestion. Elle a aussi permis la construction d'agents se déplaçant au sein de deux technologies de transports, ATM et IP, et utilisant plusieurs technologies de gestion. De plus, l'agent mobile peut utiliser plusieurs outils de gestion qui ne sont pas disponibles à distance, étendant ainsi la somme de ses moyens de gestion. Nous croyons donc que l'architecture telle qu'énoncée offre de bonnes pistes pour élaborer une architecture complète de gestion de réseaux à l'aide d'agents mobiles.

# CHAPITRE 5
# CONCLUSION

Dans ce dernier chapitre, nous présenterons d'abord une synthèse de l'architecture proposée et des agents mobiles développés pour valider cette architecture. Les points importants des travaux seront aussi résumés. Nous enchaînerons avec les limitations de l'architecture. Elles demeurent somme toute assez nombreuses et conduisent naturellement à la dernière discussion, qui porte sur les indications pour des recherches futures.

## 5.1 Synthèse des travaux

Dans ce mémoire, nous avons d'abord fait une présentation exhaustive de l'état de la recherche sur la gestion de réseaux et l'utilisation d'agents mobiles. Cette présentation a montré des applications pour quatre des cinq classes de gestion ISO. Elle a aussi mis en évidence les principales architectures de gestion et les différents modèles propres aux agents mobiles. Cette recherche a permis de proposer une architecture tirant profit des différentes conclusions de ces études et innovant en présentant des solutions pour des applications réelles sur des composantes déjà utilisées au sein des réseaux actuels.

La première étape de cet ouvrage a été de présenter une architecture de gestion de réseaux pouvant être utilisée dans des réseaux hétérogènes. L'architecture implémente la majorité de ses tâches de gestion avec un ou plusieurs agents mobiles. Elle est d'abord une série d'outils mise à la disposition des agents mobiles de gestion. Nous avons élaboré un mécanisme permettant la gestion de composantes ne pouvant pas accueillir d'agents mobiles, composantes prépondérantes dans les réseaux actuels. Notre architecture tire tout de même profit de la mobilité en tentant de gérer chaque composante le plus efficacement possible. Cette gestion est facilitée par des tables d'associations entre les composantes à gérer et les stations de gestion ayant la capacité

d'accueillir les agents. L'architecture offre aussi des interfaces de gestion permettant de simplifier la création d'agents mobiles de gestion généraux. Ces interfaces sont implémentées par des agents stationnaires qui permettent aux agents mobiles de déplacer un minimum de code. Ces agents stationnaires sont aussi dynamiques, pouvant garder leur état d'exécution et optimiser l'accès à certaines ressources distantes. Un agent stationnaire implémente une technologie de gestion et laisse l'agent mobile profiter de cette technologie par l'intermédiaire des interfaces de gestion. Nous avons pris soin de garder un maximum de flexibilité dans l'élaboration de l'architecture pour qu'elle puisse s'adapter à plusieurs réseaux et tâches de gestion.

L'architecture est effectivement prévue pour des agents mobiles. Pour éprouver les différents concepts énoncés, nous avons créé quelques agents mobiles simples chargés de tâches de gestion réelles. Le premier agent a simplement pour tâche une mise à jour des tables d'associations. Les deux autres agents mobiles sont plus complexes. L'un d'entre eux se charge de recueillir des faits sur le réseau pour éventuellement établir un diagnostic. L'autre permet de trouver de nouveaux chemins au sein d'un réseau pour aider l'agent mobile de diagnostic à déterminer les causes d'une panne encore plus précisément. Ces agents ont été éprouvés sur un réseau hétérogène utilisant deux technologies de gestion différentes (Windows et SNMP) et deux protocoles de transport différents (IP et ATM). Ils sont indépendants de la topologie et utilisent les tables de routages pour se déplacer.

Nous avons complété ce mémoire avec une étude des résultats obtenus ainsi qu'une évaluation qualitative de l'architecture en regard de ces résultats. Les principales conclusions indiquent que les agents mobiles ont su accomplir leurs tâches de gestion avec efficacité. De plus, la coopération entre deux de ces agents a démontré des capacités accrues pour la précision du diagnostic posé par l'un de ces agents. Du point de vue des performances, ils ont laissé entrevoir quelques signes qui nous laissent croire qu'ils pourraient surpasser la gestion à distance dans des scénarios plus chargés en trafic de gestion et en nombre de tâches simultanées.

## 5.2 Limitations de l'architecture

L'architecture est encore loin d'être complète. L'élaboration d'un système de gestion est un processus très complexe. Le caractère distribué des agents mobiles ne simplifie pas cette tâche. Le système de gestion implémenté a un certain nombre de limitations qu'il faudrait pallier. Tout d'abord, le nombre d'outils disponibles n'est pas complet. En effet, il n'est pas possible d'agir sur tous les aspects des réseaux. Bon nombre de paramètres ne sont pas disponibles via les interfaces de gestion. Cependant, ces aspects peuvent facilement être améliorés

Certaines limitations sont plus importantes. La sécurité en est une. L'architecture n'est pas parée aux attaques qui pourraient survenir. Bien qu'il soit simple de limiter l'accès à un client SNMP, il demeure plus difficile de distinguer les bons des mauvais agents mobiles de gestion. L'architecture fait quelques propositions en matière de sécurité, mais ces dernières pourraient s'avérer insuffisantes devant des réseaux à grande échelle et de haute importance.

La tolérance aux fautes est une autre limitation importante. Les nombreuses causes d'erreurs autant dans le réseau à gérer que le réseau de gestion sont en effet difficiles, sinon impossibles, à toutes prendre en compte. Il faudrait néanmoins améliorer cet aspect pour rendre le système plus fiable. Cette tolérance aux fautes est aussi à améliorer dans l'agent mobile de diagnostic, qui ne peut fonctionner normalement si d'éventuelles fautes surviennent pendant son parcours.

Pour rendre l'architecture plus évolutive, il faudrait permettre l'ajout et l'utilisation d'agents mobiles de gestion provenant de diverses sources. Il y a donc un certain besoin de standardisation au niveau des agents et des communications que l'architecture n'offre pas.

Les agents mobiles développés pour la validation, bien qu'ils soient complexes, demeurent toujours incapables de diagnostiquer suffisamment de pannes pour être utilisés dans un réseau de télécommunications d'importance. Ils demeurent aussi limités quant au nombre de réseaux et de technologies qu'ils peuvent appréhender. Les agents mobiles créés utilisent des réseaux à routage statique, ce qui est une limitation

importante à leur utilisation intégrale. Le diagnostic pourrait être amélioré par un mécanisme de corrélation d'erreurs plus intelligent qui pourrait, après un certain apprentissage, déterminer toujours plus de causes de pannes possibles.

D'autres recherches restent donc à faire pour améliorer à la fois l'architecture et les agents mobiles qui l'utilisent. Cela nous amène donc à indiquer quelques pistes de recherches futures.

## 5.3 Indications de recherches futures

Toutes les limitations énoncées dans la section précédentes peuvent être des indications pour des recherches futures. En effet, les recherches effectuées dans le domaine de la sécurité pourraient être appliquées à cette architecture. D'autres recherches pourraient être menées pour les problèmes de sécurité propre à l'architecture. La tolérance aux fautes est une autre voie de recherche intéressante. Les limitations au niveau de la standardisation peuvent aussi faire l'objet de recherche. Certains standards et modèles tels que FIPA, KQML et MASIF existent justement pour appréhender ces problèmes. L'architecture pourrait donc tirer profit de tels modèles. Aussi, elle pourrait être amélioré en augmentant son niveau d'automatisation et en implémentant des mécanismes de régulation des populations d'agents mobiles.

Nous avons déjà discuté des tables d'associations entre composantes à gérer et stations de gestion. Il est bon de rappeler que l'implémentation actuelle offre une association statique basée sur une proximité déterminée par le nombre de sauts. Pour un réseau de plus grande taille, il pourrait être intéressant de créer un algorithme permettant, selon les différentes données sur les vitesses des liens, des capacités des composantes et/ou tout autre paramètre, de déterminer la meilleure façon d'associer les différentes stations de gestion aux différentes composantes. Suite à l'élaboration d'un tel algorithme, il serait intéressant d'implémenter un mécanisme de mise à jour automatique qui évaluerait, un peu comme RIP (Routing Information Protocol) le fait avec les tables de routages, la meilleure disposition selon les modifications sur la topologie ou le comportement du réseau. Les travaux de Liotta et al. (2002) offrent

déjà quelques pistes pour créer un tel algorithme. Les tables d'associations pourraient être initialisées par un agent mobile chargé de découvrir les différentes composantes du réseau ainsi que les stations de gestion. Un tel agent pourrait être appuyé par l'algorithme discuté.

Du côté de l'agent mobile de diagnostic, plusieurs recherches pourraient être entreprises. Il faudrait améliorer sa précision et la confiance en ses conclusions. Cette confiance pourrait éventuellement permettre l'élaboration non pas seulement d'agents mobiles de diagnostic, mais aussi d'agents mobiles réparateurs. Les agents mobiles de recherche pourraient aussi avertir l'agent mobile de diagnostic lorsqu'aucun chemin n'est trouvé. Les agents mobiles de diagnostic et de recherche pourraient être reconstruits avec comme capacité l'utilisation de tables de routage dynamiques. Ils pourraient aussi avoir la possibilité de trouver les causes de pannes multiples.

Finalement, l'architecture pourrait être enrichie par d'autres agents mobiles de gestion implémentant d'autres tâches de gestion. La revue de littérature regorge d'applications diverses pour les agents mobiles. La gestion de pannes n'était qu'une application parmi d'autres possibles. Elle n'en demeure pas moins l'une des disciplines importantes de la gestion de réseaux.

# BIBLIOGRAPHIE

AGENTBUILDER. Agent Construction Tools: Commercial Products. *In Site de* **AgentBuilder** [En ligne]. http://www.agentbuilder.com/AgentTools/commercial.php. (Page consultée le 11 juin 2002).

AMIN, K.A., MAYES, J.T., MIKLER, A.R. "Agent-based distance vector routing". *Mobile Agents for Telecommunication Applications. Third International Workshop, MATA 2001. Proceedings of MATA'01. 3rd International Workshop on Mobile Agents for Telecommunication Applications, 14-16 août 2001, Montreal, Canada* , pp. 41-50.

APPLEBY, S., STEWARD, S. « Mobile software agents for control in telecommunications networks », *British Telecom Technology Journal*, vol. 12, no. 2, avril 1994, pp. 104-113.

ASAKA, M., ONABURA, T., INOUE, T., GOTO, S. "Remote attack detection method in IDA: MLSI-based intrusion detection using discriminant analysis". *Proceedings 2002 Symposium on Applications and the Internet (SAINT 2002), Proceedings 2002 Symposium on Applications and the Internet (SAINT 2002), 28 Janvier -1$^{er}$ février 2002, Nara, Japon,* pp. 64-73.

ASAKA, M., TAGUCHI, A., GOTO, S. "The Implementation of IDA: An Intrusion Detection Agent System". *Proceedings of the 11th Annual FIRST Conference on Computer Security Incident Handling and Response (FIRST'99), 13-18 juin 1999, Brisbane, Australia.*

BALDI, M., GAI, S., PICCO, G. P. "Exploiting Code Mobility in decentralized and Flexible Network Management". *Mobile Agents. First International Workshop, MA '97 Proceedings, Mobile Agents. First International Workshop, MA '97. Proceedings*, 7-8 avril 1997, *Berlin, Germany*, pp. 13-26.

BARÁN, B., SOSA, R. "A new approach for AntNet routing". *Engbersen, T., Park, E.K., Proceedings Ninth International Conference on Computer Communications and Networks, Proceedings Ninth International Conference on Computer Communications and Networks, 16-18 Oct. 2000, Las Vegas, NV, USA,* pp. 303-308.

BERNARDES, M.C., DOS SANTOS MOREIRA, E. "Implementation of an intrusion detection system based on mobile agents". *Proceedings International Symposium on Software Engineering for Parallel and Distributed Systems, Proceedings of 5th International Symposium on Software Engineering for Parallel and Distributed Systems, 10-11 juin 2000, Limerick, Ireland*, pp. 158-164.

BIESZCZAD, A., PAGUREK, B., WHITE, T. "Mobile Agents for Network Management", *IEEE Communications Survey*. vol. 1, no. 1, 1998.

BIESZCZAD, A., BISWAS, P. K., BUGA, W., MALEK, M., TAN, H. "Management of Heterogeneous Networks with Intelligent Agents". *Bell Labs Technical Journal*. vol. 4, no. 4, oct.-déc. 1999, pp. 109-35

BOYER, J., PAGUREK, B., WHITE, T. "Methodologies for PVC Configuration in Heterogeneous ATM Environment Using Intelligent Mobile Agents". *Proceedings of MATA '99, Mobile Agents for Telecommunications Applications, Ottawa, Canada, Oct. 1999.*

BRUSIC, I., HASSLER, V., LUGMAYR, W. "Deployment of Mobile Agents in the Mobile Telephone Network Management". *Proceedings of the 33rd Annual Hawaii International Conference on Systems Sciences, Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, 4-7 Jan. 2000, Maui, HI, USA,* 9 p.

CHANG, F-M., KAO, S-J. "A hierarchically dynamic network management system using mobile agents". *IEEE Intelligent Network 2001 Workshop. IN 2001 Conference Record, IEEE Intelligent Network 2001 Workshop. IN 2001 Conference Record, 6-9 mai 2001, Boston, MA, USA,* pp. 130-134.

CHEIKHROUHOU, M., CONTI, P., LABETOULLE, J. "Automatic Configuration of PVCs in ATM Networks with Software Agents". *Hong, J.W., Weihmayer, R., NOMS 2000. 2000 IEEE/IFIP Network Operations and Management Symposium 'The Networked Planet: Management Beyond 2000', Proceedings of Network Operations and Management Symposium, 10-14 avril 2000, Honolulu, HI, USA,* pp. 535-548.

CHPUDHURY, R.R., BANDYOPADHYAY, S. PAUL., K., "A distributed mechanism for topology discovery in ad hoc wireless networks using mobile agents". *2000 First Annual Workshop on Mobile and Ad Hoc Networking and Computing. MobiHOC, Proceedings of First Annual Workshop on Mobile Ad Hoc Networking Computing. MobiHOC Mobile Ad Hoc Networking and Computing, 11 août 2000,* pp. 145-146.

DASGUPTA, D., BRIAN, H. "Mobile security agents for network traffic analysis". *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01, vol.2 , 2001, Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01, 12-14 juin 2001, Anaheim, CA, USA,* pp. 332-340.

DI CARO, G., DORIGO, M. "Ant colonies for adaptive routing in packet-switched communications networks". *Parallel Problem Solving from Nature - PPSN V. 5th International Conference. Proceedings, Parallel Problem Solving from Nature - PPSN V. 5th International Conference. Proceedings, 27-30 sept. 1998, Amsterdam, Netherlands,* pp. 673-682.

EL-DARIEBY, M., BIESZCZAD, A. "Intelligent mobile agents: towards network fault management automation". *Integrated Network Management VI. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, 24-28 mai 1999, Boston, MA, USA,* pp. 611-222.

FOUKIA, N., HULAAS, J., HARMS, J. "Intrusion Detection with Mobile Agents". *proceedings of the 11th Annual Internet Society Conference (INET 2001), Stockholm, Sweden, 5-8 juin 2001.*

GAVALAS, D., GREENWOOD, D., GHANBARI, M., O'MAHONY, M. "Hierarchical network management: a scalable and dynamic mobile agent-based approach". *Computer Networks.* vol. 38, no. 6, april 2002, pp. 693-711.

GAVALAS, D., GREENWOOD, D., GHANBARI, M., O'MAHONY, M, "Advanced network monitoring applications based on mobile/intelligent agent technology", Computer Communications. vol. 23, no. 8, april 2000, pp. 720-30.

GÜNTER, M., BRAUN, T. "Internet Service Monitoring with Mobile Agents". *IEEE Network* vol. 16, no. 3, mai-juin 2002, pp. 22-29.

HALL, R. S., HEIMBIGNER, D., AND WOLF, A. L. "A Cooperative Approach to Support Software Deployment Using the Software Dock". *Proceedings of the 21st International Conference on Software Engineering (ICSE '99), 16-22 mai 1999, Los Angeles, CA, USA,* pp. 174-183.

HELMER, G., WONG, J.S.K., HONAVAR, V., MILLER, L. "Automated discovery of concise predictive rules for intrusion detection". *Journal of Systems and Software,* vol. 60, no. 3, 15 février 2002, pp. 165-75.

HUSTON, G. *ISP Survival Guide: Strategies for Running a Competitive ISP*, John Wiley & Sons, 1999, 661p.

HWANG, K. GANGADHARAN, M. "Micro-firewalls for dynamic network security with distributed intrusion detection". *Proceedings IEEE International Symposium on Network Computing and Applications. NCA 2001. NCA 2001, 8-10 oct. 2001, Cambridge, MA, USA*, pp. 68-79.

KIM, S.-T., PARK, H.-J., KIM, Y.-C. "The Load Monitoring of Web Server using Mobile Agent". *2001 International Conferences on Info-Tech and Info-Net. 29 oct.-1$^{er}$ nov. 2001, Beijing, China,* vol. 6, 130-105.

KNIGHT, G., HAZEMI, R., "Mobile Agent-Based Management in the INSERT Project". *Journal of Network and Systems Management.* vol. 7, no. 3, sept. 1999, pp. 271-93.

KONA, M. K., XU, C-Z. "A Framework for Network Management using Mobile Agents". *Proceedings of the First IEEE Int'l Workshop on Internet Computing and E-Commerce, San Francisco, avril, 2001.*

KRÜGEL, C., TOTH, T., KIRDA, E. "SPARTA-a mobile agent based intrusion detection system". *Advances in Network and Distributed Systems Security. IFIP TC1 WG11.4. First Annual Working Conference on Network Security. 26-27 nov. 2001, Leuven, Belgium*, pp. 187-198.

LA CORTE, A., PULIAFITO, A.,TOMARCHIO, O. "QoS management in programmable networks through mobile agents", *Microprocessors and Microsystems,* vol. 25, no. 2, avril 2001, pp. 111-20.

LAZAR, S., SIDHU, D., KODESWARAN, S., VARADHARAJ, R. "ATM Network Discovery Using Mobile Agents", *Proceedings 24th Conference on Local Computer Networks. LCN'99, 18-20 oct. 1999, Lowell, MA, USA,* pp. 98-105.

LIOTTA, A., PAVLOU, G., KNIGHT, G. "Exploiting Agent Mobility for Large-Scale Network Monitoring". *IEEE Network.* vol. 16, no. 3, mai-juin 2002, pp. 7-15.

LIOTTA, A., KNIGHT, G., PAVLOU, G. "Modelling Network and System Monitoring Over the Internet with Mobile Agents". *NOMS 98. 1998 IEEE Network Operations and Management Symposium. Conference Proceedings, vol.2, 1998, février 1998, New Orleans, LA, USA,* vol. 2, pp. 303-312.

LIPPERTS, S. "Enabling Alarm Correlation for a Mobile Agent Based System and Network Management – A Wrapper Concept". *Proceedings of ICON'99: IEEE International Conference on Networks, 28 sept.-1er oct. 1999, Brisbane, Qld., Australia,* pp. 125-132.

MANVI, S.S., VENKATARAM, P. "QoS Management By Mobile Agents in Multimedia Communication". *Proceedings 11th International Workshop on Database and Expert Systems Applications, 4-8 Sept. 2000, London, UK,* pp. 407-411.

MELL, P., MCLARNON, M. "Mobile Agent Attack Resistant Distributed Hierarchical Intrusion Detection Systems". RAID99, 1999, [En ligne] http://www.raid-symposium.org/raid99/PAPERS/Mell.pdf (Page consultée le 21 juin 2002).

MELL, P., MARKS, D., MCLARNON, M. "A denial-of-service resistant intrusion detection architecture". *Computer Networks,* vol. 34, no. 4, oct. 2000, pp. 641-58.

ONISHI, R., YAMAGUCHI, S., MORINO, H., AIDA, H., SAITO, T. "The Multi-agent System for Dynamic Network Routing". *Proceedings of 5th International Symposium on Autonomous Decentralized Systems, 26-28 mars 2001, Dallas, TX, USA*, pp. 375-382.

PAGUREK, B., LI, Y., BIESZCZAD, A., SUSILO, G. "Network Configuration Management in Heterogeneous ATM Environments". *Second International Workshop, IATA'98 Proceedings, 4-7 juillet 1998, Paris, France,* pp. 72-88.

PINHEIRO, R., POYLISHER, A., CALDWELL, H. "Mobile Agents for Aggregation of Network Management Data". *First International Symposium on Agent Systems and Applications/Third International Symposium on Mobile Agents, 3-6 oct. 1999, Palm Springs, CA, USA*, pp. 130-140.

PULIAFITO, A., TOMARCHIO, O. "Using mobile agents to implement flexible network management strategies". *Computer Communications.* vol. 23, no. 8, april 2000, pp. 708-19.

PULIAFITO, A., TOMARCHIO, O. "Advanced network management functionalities through the use of mobile software agents". Albayrak, S., Intelligent Agents for Telecommunication Applications. Third International Workshop, IATA'99. Proceedings, Berlin, Germany : Springer-Verlag, 1999, pp. 33-45.

PUTZOLU, D., BAKSHI, S., YADAV, S., YAVATKAR, R. "The Phoenix Framework: A Practical Architecture for Programmable Networks". *IEEE Communications,* vol. 38, no. 3, mars 2000, pp. 160-5.

RUBINSTEIN, M. G., DUARTE, O. C. M. N., PUJOLLE, G. "Scalability of a Network Management Application Based on Mobile Agents", March 2002. (Technical report), [URL] http://www.gta.ufrj.br/~rubi/public.html, 13 juin 2002.

RUBINSTEIN, M. G., DUARTE, O. C. M. N., PUJOLLE, G. "Reducing the Response Time in Network Management by Using Multiple Mobile Agents", In: de Souza, J.N., Boutaba, R., Managing QoS in Multimedia Networks and Services, IEEE/IFIP TC6-WG6.4 & WG6.6. Third International Conference on Management of Multimedia Networks and Services (MMNS'2000), Norwell, MA, USA : Kluwer Academic Publishers, 2000, pp. 253-265.

SILVA, L. M., SIMÕES, P., SOARES, G., MARTINS, P., BATISTA, V., RENATO, C., ALMEIDA, L., STOHR, N. "JAMES: A Platform of Mobile Agents for the Management of Telecommunication Networks". *Third International Workshop, IATA'99. Proceedings, 9-10 août. 1999, Stockholm, Sweden*, pp. 77-95. (a)

SILVA, L. M., SOARES, G., MARTINS, P. "The performance of mobile agent platforms". *Third International Symposium on Mobile Agents, 3-6 oct. 1999, Palm Springs, CA, USA,* pp. 270-271. (b)

STOREY, M., BLAIR, G. "Resource Configuration in Ad Hoc Networks: The MARE Approach". *Proceedings Third IEEE Workshop on Mobile Computing Systems and Applications, 7-8 déc. 2000, Los Alamitos, CA, USA*, pp. 60-69.

SUGAR, R., IMRE, S. "Adaptive clustering using mobile agents in wireless ad-hoc networks". *Lecture Notes in Computer Science vol.2158, Proceedings of IDMS'01. Interactive Distributed Multimedia Systems, 4-7 sept. 2001, Lancaster, UK*, pp. 199-204.

SURI, N., GROTH, P.T,, BRADSHAW, J.M. "While You're Away: a system for load-balancing and resource sharing based on mobile agents". *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid, 15-18 mai 2001, Brisbane, Qld., Australia,* pp. 470-473.

SUSILO, G., BIESZCZAD, A., PAGUREK, B. "Infrastructure for Advanced Network Management based on Mobile Code". *NOMS 98 1998 IEEE Network Operations and Management Symposium, 15-20 février 1998, New Orleans, LA, USA*, vol. 2, pp. 322-333.

TOMARCHIO, O., VITA, L. "On the use of mobile code technology for monitoring Grid system". *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid, 15-18 May 2001, Brisbane, Qld., Australia*, pp. 450-455.

WHITE, T., PAGUREK, B., BIESZCZAD, A. "Network Modeling for Management Applications Using Intelligent Mobile Agents". *Journal of Network and Systems Management,* vol. 7, no. 3, sept. 1999, pp. 295-321.

WHITE, T., PAGUREK, B. "Distributed fault location in networks using learning mobile agents". *Lecture Notes in Artificial Intelligence, vol.1733, Second Pacific Rim International Workshop on Multi-Agents, PRIMA'99, 2-3 déc. 1999, Kyoto, Japan,* pp. 182-196.

WHITE, T., BIESZCZAD, A., PAGUREK, B. "Distributed fault location in networks using mobile agents". *Second International Workshop, IATA'98 Proceedings, 4-7 juillet 1998, Paris, France,* pp. 130-141.

ZHANG, P., SUN, Y. "A New Approach Based on Mobile Agents to Network Fault Detection". *Proceedings 2001 International Conference on Computer Networks and Mobile Computing, 16-19 Oct. 2001, Los Alamitos, CA, USA,* pp. 229-234.

# ANNEXE A

# CONFIGURATIONS POUR LE RÉSEAU TRIANGLE

Commutateur ATM – Fidji

```
!
version 12.0
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Fidji
!
ip subnet-zero
no ip domain-lookup
!
atm address XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXx
atm router pnni
 no aesa embedded-number left-justified
 node 1 level 56 lowest
  redistribute atm-static
!
!
!
interface ATM0/0/0
 no ip address
 no ip directed-broadcast
!
interface ATM0/0/1
 no ip address
 no ip directed-broadcast
 atm pvc 1 50  interface  ATM0/0/0 1 33
!
interface ATM0
 ip address 192.168.10.100 255.255.255.0
 no ip directed-broadcast
 map-group test
 atm maxvp-number 0
 atm pvc 0 100  encap aal5snap interface  ATM0/0/0 1 100
 atm pvc 0 101  encap aal5snap interface  ATM0/0/1 1 101
!
!
ip classless
ip route 192.168.3.0 255.255.255.0 192.168.10.1
ip route 192.168.4.0 255.255.255.0 192.168.10.1
ip route 192.168.5.0 255.255.255.0 192.168.10.3
ip route 192.168.50.0 255.255.255.0 192.168.10.1
ip route 192.168.51.0 255.255.255.0 192.168.10.1
!
!
```

```
map-list test
 ip 192.168.10.1 atm-vc 101
 ip 192.168.10.3 atm-vc 100
snmp-server community public RO
!
end
```

Routeur - Montréal

```
!
version 12.2
!
hostname Montreal
!
fax interface-type fax-mail
mta receive maximum-recipients 0
!
interface Ethernet0/0
 ip address 192.168.50.2 255.255.255.0
 no ip mroute-cache
 half-duplex
 no cdp enable
!
interface FastEthernet1/0
 ip address 192.168.3.1 255.255.255.0
 no ip mroute-cache
 speed 100
 full-duplex
 no cdp enable
!
interface ATM3/0
 no ip address
 no ip mroute-cache
 no atm scrambling cell-payload
 atm ilmi-keepalive
!
interface ATM3/0.1 point-to-point
 ip address 192.168.11.1 255.255.255.0
 pvc chemin 1/50
  protocol ip 192.168.11.2 broadcast
  ubr 155000
  encapsulation aal5snap
 !
!
interface ATM3/0.2 point-to-point
 ip address 192.168.10.1 255.255.255.0
 pvc chemin2 1/101
  protocol ip 192.168.10.2
  ubr 155000
  encapsulation aal5snap
 !
!
no ip classless
ip route 192.168.4.0 255.255.255.0 192.168.50.1
ip route 192.168.5.0 255.255.255.0 ATM3/0.1
ip route 192.168.10.0 255.255.255.0 ATM3/0.2
```

```
ip route 192.168.11.0 255.255.255.0 ATM3/0.1
ip route 192.168.51.0 255.255.255.0 192.168.50.1
no ip http server
!
!
snmp-server community public RO
call rsvp-sync
!
end
```

Routeur - Vancouver

```
!
version 12.1
no service single-slot-reload-enable
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Vancouver
!
ip subnet-zero
!
!
no ip finger
no ip domain-lookup
!
call rsvp-sync
voice rtp send-recv
!
interface Ethernet0/0
 ip address 192.168.51.1 255.255.255.0
 half-duplex
!
interface Ethernet0/1
 ip address 192.168.50.1 255.255.255.0
 full-duplex
 h323-gateway voip interface
!
interface FastEthernet1/0
 ip address 192.168.4.1 255.255.255.0
 speed 100
 full-duplex
!
no ip classless
ip route 192.168.3.0 255.255.255.0 192.168.50.2
ip route 192.168.5.0 255.255.255.0 192.168.51.2
ip route 192.168.10.0 255.255.255.0 192.168.50.2
no ip http server
!
snmp-server community public RO
!
end
```

# ANNEXE B

# CONFIGURATIONS POUR LE RÉSEAU LINÉAIRE

Routeur - Boston

```
!
version 12.2
!
hostname Boston
!
fax interface-type fax-mail
mta receive maximum-recipients 0
!
interface Ethernet0/0
 ip address 192.168.51.2 255.255.255.0
 no ip mroute-cache
 half-duplex
 no cdp enable
!
interface FastEthernet1/0
 ip address 192.168.5.1 255.255.255.0
 no ip mroute-cache
 speed 100
 full-duplex
 no cdp enable
!
interface ATM3/0
 no ip address
 no ip mroute-cache
 no atm scrambling cell-payload
 atm ilmi-keepalive
!
interface ATM3/0.1 point-to-point
 ip address 192.168.12.1 255.255.255.0
 pvc chemin 1/33
  protocol ip 192.168.12.2 broadcast
  ubr 155000
  encapsulation aal5snap
 !
!
interface ATM3/0.2 point-to-point
 ip address 192.168.10.3 255.255.255.0
 pvc chemin2 1/100
  protocol ip 192.168.10.4
  ubr 155000
  encapsulation aal5snap
 !
!
no ip classless
ip route 192.168.4.0 255.255.255.0 192.168.51.1
ip route 192.168.3.0 255.255.255.0 ATM3/0.1
ip route 192.168.10.0 255.255.255.0 ATM3/0.2
ip route 192.168.11.0 255.255.255.0 ATM3/0.1
```

```
ip route 192.168.50.0 255.255.255.0 192.168.51.1
no ip http server
!
snmp-server community public RO
!end
```

Routeur - Montréal

```
!
version 12.2
!
hostname Montreal
!
fax interface-type fax-mail
mta receive maximum-recipients 0
!
interface Ethernet0/0
 ip address 192.168.50.2 255.255.255.0
 no ip mroute-cache
 half-duplex
 no cdp enable
!
interface FastEthernet1/0
 ip address 192.168.3.1 255.255.255.0
 no ip mroute-cache
 speed 100
 full-duplex
 no cdp enable
!
no ip classless
ip route 192.168.4.0 255.255.255.0 192.168.50.1
ip route 192.168.5.0 255.255.255.0 192.168.50.1
ip route 192.168.51.0 255.255.255.0 192.168.50.1
no ip http server
!
!
snmp-server community public RO
call rsvp-sync
!
end
```

Routeur - Vancouver

```
!
version 12.1
no service single-slot-reload-enable
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Vancouver
!
ip subnet-zero
```

```
!
!
no ip finger
no ip domain-lookup
!
call rsvp-sync
voice rtp send-recv
!
interface Ethernet0/0
 ip address 192.168.51.1 255.255.255.0
 half-duplex
!
interface Ethernet0/1
 ip address 192.168.50.1 255.255.255.0
 full-duplex
 h323-gateway voip interface
!
interface FastEthernet1/0
 ip address 192.168.4.1 255.255.255.0
 speed 100
 full-duplex
!
no ip classless
ip route 192.168.3.0 255.255.255.0 192.168.50.2
ip route 192.168.5.0 255.255.255.0 192.168.51.2
no ip http server
!
snmp-server community public RO
!
end
```

Routeur - Boston

```
!
version 12.2
!
hostname Boston
!
fax interface-type fax-mail
mta receive maximum-recipients 0
!
interface Ethernet0/0
 ip address 192.168.51.2 255.255.255.0
 no ip mroute-cache
 half-duplex
 no cdp enable
!
interface FastEthernet1/0
 ip address 192.168.5.1 255.255.255.0
 no ip mroute-cache
 speed 100
 full-duplex
 no cdp enable
!
no ip classless
ip route 192.168.4.0 255.255.255.0 192.168.51.1
ip route 192.168.3.0 255.255.255.0 192.168.51.1
```

```
ip route 192.168.50.0 255.255.255.0 192.168.51.1
no ip http server
!
snmp-server community public RO
call rsvp-sync
!
end
```